Rendimiento de algoritmos de reconocimiento facial para autenticación biométrica en sistemas de monitoreo doméstico con Edge Computing

.

Melby Vanessa Luna Rosero
Gerardo Alejandro Muñoz López

Universidad Mariana
Facultad de Ingeniería
Ingeniería Mecatrónica
San Juan de Pasto
2024

Rendimiento de algoritmos de reconocimiento facial para autenticación biométrica en sistemas de monitoreo doméstico con Edge Computing

Melby Vanessa Luna Rosero
Gerardo Alejandro Muñoz López

Informe de investigación para optar al título de: Ingeniero(s) Mecatrónico

Asesor: M.Sc. Fausto Andrés Escobar Revelo
Coasesor: PhD(c). Dagoberto Mayorca Torres

Universidad Mariana
Facultad de Ingeniería
Ingeniería Mecatrónica
San Juan de Pasto
2024

Artículo 71: los conceptos, afirmaciones y opiniones emitidos en el Trabajo de Grado son responsabilidad única y exclusiva del (los) Educando (s)

Reglamento de Investigaciones y Publicaciones, 2007
Universidad Mariana

Contenido

Pág.

Índice de Tablas

Pág.

Índice de figuras

Pág.

Índice de ecuaciones

Pág.

Índice de anexos

Pág.

1. Resumen del proyecto

Este trabajo de grado presenta una investigación centrada en el análisis comparativo de técnicas de reconocimiento facial, aplicando enfoques supervisados de aprendizaje automático en una plataforma de Edge Computing. Se seleccionó la tarjeta Jetson Nano de Nvidia por su capacidad de procesamiento eficiente, crucial para la ejecución en tiempo real de modelos complejos.

Actualmente, se emplean diversas técnicas, métodos y algoritmos para el reconocimiento facial, cada uno con características y métricas distintas que evalúan su rendimiento. Sin embargo, estos también presentan desafíos, como la transmisión de imágenes capturadas en condiciones de baja iluminación, sesgos raciales, poses, el paso del tiempo y el uso de elementos faciales.

Este estudio se enfoca en la evaluación de métricas clave como precisión, confianza, *Recall* y la matriz de confusión de los algoritmos seleccionados, específicamente YOLOv8, VGG y ResNet50, ejecutados utilizando frameworks como PyTorch. La base de datos utilizada, compuesta por aproximadamente 893 imágenes, fue dividida en una proporción de 74/26 para entrenamiento y validación. Se aplicaron técnicas de aumento de datos para optimizar el rendimiento de los modelos entrenados.

Los resultados muestran que el modelo más robusto alcanzó una precisión superior al 90% y una confianza del 89%, con un tiempo de ejecución menor en comparación con los otros dos modelos, lo que lo hace adecuado para su uso en sistemas de monitoreo residencial bajo condiciones óptimas.

1.1. Descripción del problema

Actualmente, se utilizan una amplia variedad de algoritmos para el procesamiento de imágenes y el reconocimiento facial en campos como la seguridad, la medicina diagnóstica, el entrenamiento y las aplicaciones militares. El objetivo principal es lograr una inferencia más eficiente y precisa, reduciendo el coste computacional, especialmente en plataformas móviles o de Edge Computing para la identificación de rostros (Diego, 2002). Además, el uso de sistemas informáticos

combinados con aprendizaje automático, especialmente a través de redes neuronales profundas, ha permitido avances significativos en el reconocimiento facial y en la parametrización de un espacio euclidiano para medir las distancias entre características faciales.

Sin embargo, diversos factores pueden influir negativamente en la detección y autenticación facial. Entre ellos se encuentran el uso inadecuado de bases de datos de entrenamiento ya que al crear una propia base de datos se deben tener en cuenta el tamaño de las imágenes ,su iluminación, calidad y las poses (Insaf-Adjabi, 2020) , el procesamiento incorrecto de la iluminación en la distribución facial, los cambios de perspectiva en vistas laterales 2D y los sesgos derivados del origen racial o étnico, como las diferencias en el color de piel o los rasgos faciales característicos de ciertas poblaciones. Estos factores pueden causar problemas de precisión y afectar la seguridad al generar falsos positivos durante el proceso de autenticación.

La implementación de sistemas de monitoreo que incorporan tecnología de reconocimiento facial enfrenta un desafío significativo relacionado con el coste. Los sistemas disponibles en el mercado suelen ser costosos porque requieren la adquisición de cámaras especializadas con capacidad de reconocimiento facial. Estos sistemas, además, necesitan dispositivos computacionales avanzados para procesar grandes volúmenes de datos en tiempo real, lo que implica una inversión considerable en hardware especializado, como cámaras de alta resolución y servidores. Sin embargo, una solución más rentable es aplicar esta tecnología en una plataforma de Edge Computing, que optimiza los recursos y reduce la necesidad de hardware costoso, manteniendo al mismo tiempo la precisión y eficiencia necesarias para la identificación de rostros en el monitoreo de seguridad (Lasse Rouhiainen, 2018).

Por tanto, es pertinente investigar, comparar y evaluar los algoritmos con respecto al desempeño en factores como el tiempo de procesamiento, la precisión y la confianza en el uso del algoritmo en la plataforma de *Edge Computing* y recursos *Hardware*, como la GPU y memoria en el proceso de autenticación facial; es así como este proyecto pretende realizar un estudio comparativo de modelos basado en el reconocimiento facial y aprendizaje automático en una plataforma de *Edge Computing*, lo cual plantea el siguiente interrogante.

1.1.1. Formulación del problema

¿Cuál es el algoritmo más óptimo para la autenticación biométrica basado en el reconocimiento facial y aprendizaje automático en una plataforma de Edge *Computing*?

1.2. Justificación

La creciente adopción de tecnologías de reconocimiento facial en ámbitos como la seguridad y el monitoreo del hogar subraya la necesidad de desarrollar soluciones más eficientes y accesibles. Este proyecto se propone contribuir de manera innovadora al campo del reconocimiento facial en plataformas de Edge Computing, abordando brechas técnicas significativas y alineándose con las tendencias actuales en la investigación y aplicación de estas tecnologías.

El uso de plataformas de Edge Computing, como la Jetson Nano, presenta una oportunidad para optimizar el procesamiento de imágenes y la identificación facial en tiempo real, reduciendo la dependencia de infraestructuras costosas y mejorando la eficiencia computacional. Sin embargo, uno de los desafíos más importantes es la adaptación de algoritmos de reconocimiento facial a las limitaciones de hardware y las condiciones de operación en entornos no controlados. Este proyecto busca cerrar estas brechas técnicas, proporcionando una comparativa de tres modelos que mantengan altos niveles de precisión y rendimiento, incluso bajo condiciones de iluminación variable, poses faciales diversas y video en tiempo real. Shiguang Shan, 2012

La investigación se justifica no solo por su potencial para mejorar las capacidades técnicas de los sistemas de monitoreo del hogar, sino también por su contribución a la reducción de costes y el aumento de la accesibilidad de estas tecnologías. Al analizar y comparar tres modelos de aprendizaje automático para el reconocimiento facial, este proyecto no solo evalúa el rendimiento en términos de precisión, tiempo de procesamiento y eficiencia, sino que también aborda la necesidad de modelos robustos que puedan ser implementados en escenarios reales con recursos limitados.

1.3. Objetivos

1.3.1 Objetivo General

Estudio comparativo de algoritmos de autenticación biométrica basado en el reconocimiento facial y aprendizaje automático en una plataforma de Edge *Computing.*

1.3.2 Objetivos específicos

Identificar y analizar los criterios de autentificación biométrica y aprendizaje automático basado en reconocimiento de rostros.

Evaluar y comparar diferentes algoritmos de identificación, extracción y clasificación de características para la detección de rostros en video basado en métodos de aprendizaje automático supervisado.

Realizar una evaluación estadística de los algoritmos en una plataforma de *Edge Computing* a través de un conjunto de datos para la identificación de rostros, considerando factores como la precisión, robustez y repetibilidad del sistema.

1.4 Marco referencial o fundamentos teóricos

1.4.1 *Antecedentes*

En la actualidad el uso de técnicas de aprendizaje automático para el reconocimiento facial ha traído cambios significativos a la tecnología de seguridad doméstica debido a su capacidad para identificar y autenticar a las personas mediante el análisis de sus características únicas en cada rostro. Por ejemplo, en China se emplean sistemas de seguridad para controlar la seguridad de los habitantes mediante la extracción de características faciales, mientras que en los países

latinoamericanos se emplean sistemas de seguridad estándar mediante cámaras y alarmas sonoras que se activan mediante un sensor.

1.4.1.1 Base de datos y criterios de búsqueda. Con la finalidad de lograr los resultados deseados, a continuación, se presenta una metodología de mapeo sistemático, que permite identificar y comprender los lineamientos para abordar el desarrollo del proyecto. La metodología emplea una serie de preguntas de indagación, búsqueda de literatura relacionada a la temática, extracción de datos relevantes y finalmente un análisis y clasificación de la literatura. Para realizar la búsqueda, se emplearon las bases de datos científicas IEEEXplore y ScienceDirect. Los criterios de búsqueda y selección de artículos son: artículos más citados y de mayor relevancia en los últimos 5 años, esta investigación se realizó en mayo del 2023 enfocándonos en artículos de seguridad domótica, algoritmos de procesamiento de video y aprendizaje automático, un ejemplo se puede observar en la Tabla 1.

Tabla 1.

Criterios de búsqueda enfocada a reconocimiento facial pose e iluminación

Tipo de búsqueda	Año
Criterios de búsqueda	"Algorithm"AND "Image" AND "Face" AND Recognition" "haar cascade"AND "face" AND "Recognition" AND "Neural Network" AND "Classifier" AND "Image" AND "artificial intelligence" AND "Techniques" AND "Classifier"
Periodo de búsqueda	2017-2023
Número de documentos encontrados sin filtros	71
Idioma	inglés
Filtro por área temática	Ingeniería Inteligencia artificial Computación

Tipo de documentos	Artículos
Número de documentos encontrados con filtros	20

Descripción de los estudios. Los aspectos más destacados durante la búsqueda se observan en la Tabla 2, teniendo en cuenta las citaciones realizadas para los artículos que mejor resultados obtuvieron en la detección de reconocimiento de rostros y factores que influyen como cambios de pose y luminosidad.

Tabla 2.

Base de datos artículos que mejor resultado obtuvieron

N	Año	Autores	Número de citaciones	Área temática
1	2021	Anirudha B Shetty Bhoomika Deek sha Jeevan Rebeiro Ram yashree	10	Facial recognition using Haar cascade and LBP classifiers (reconocimiento facial)
2	2022	vítor garabeto, tiago cruz, Paulo Simöes	1	Security of Building Automation and Control Systems: Survey and future research directions
3	2017	QingshanLiu, Jing Yang, JiankangDeng ,KaihuaZhang	21	Robust facial landmark tracking via cascade regression
4	2018	VinayAAviralJoshiHardi kMahipal SuranaHarshGargK	1	Unconstrained Face Recognition using ASURF and Cloud-Forest Classifier optimized with VLAD

		NBalasubramanya MurthySNatarajan		
5	2018	Zhang NaNa ZhangJinb	11	Optimization of Face Tracking Based on KCF and Camshift

1.4.1.2 Descripción de los artículos. El uso de seguridad domótica y algoritmos de procesamiento de imágenes Tabla 2 se evidencian diferentes problemas en cuanto a iluminación en interiores y exteriores, pose, orientación del rostro, ruido instrumental, expresión facial, oclusión debido a objetos o accesorios tales como gafas de sol, sombreros, vello facial y envejecimiento (Vinay et al., 2018). Lo que implica un inconveniente en la búsqueda de un algoritmo capaz de realizar reconocimiento facial en la identificación del rostro.

Diversas investigaciones se han centrado en la descripción de la seguridad domótica y los algoritmos empleados para realizar el reconocimiento facial. En el artículo de (Anirudha-Shetty et al., 2021), ponen a discusión y prueba dos algoritmos Haarcascade y Local Binary Pattern (LBP) utilizados usualmente para la clasificación de rostros de infantes indicando que teniendo en cuenta factores como: claridad de imagen, pose recta del rostro, los cuales pueden influir en la precisión del reconocimiento facial. También indica que el algoritmo basado en Haarcascade es más preciso en un 95% (UniPython, 2021) que el LBP al identificar personas adultas, pero tiene una falla en la detección de rostros en niños.

En cuanto a los sistemas de control y automatización empleados en los sistemas domóticos en edificios (BACS) (Paulo-Simöes et al., 2022), indica que a pesar de la evolución tecnológica encaminadas a la seguridad como la comunicación encriptada todavía es propensa a recibir ataques a su seguridad puesto que esta todavía es muy primitiva y no le han dado la suficiente importancia dando por resultado a un foco de ataques ya sea en la manipulación física de los sensores, actuadores o directamente a su programación. Por tanto, aún es un campo de investigación que plantea posibles soluciones en la mejora de la seguridad y atacar las vulnerabilidades en el sistema

de seguridad domótico en cuanto a el reconocimiento facial y pose, mediante la optimización con el fin de tener un sistema robusto y difícil de quebrantar.

Las mejoras que han existido desde 2017 sobre las localizaciones de puntos faciales en imágenes estáticas, mejorando la toma de imágenes secuenciales, pose e iluminación, empleando un sistema de regresión en cascada el cual es capaz de seguir y predecir la ubicación de los puntos faciales disminuyendo variaciones. Para realizar el proceso de aprendizaje se emplea un modelo de regresión más robusto frente a cambio de variaciones en la pose que podrían existir en una imagen secuencial; este modelo podría ubicar de manera más precisa puntos faciales del rostro cuando esta se encuentre borrosa (Kaihua Zhan et al., 2017).

De igual manera en el documento de Vinay A et al. (2018) señala la importancia de emplear los algoritmos Haar-Cascade, Affine Speeded-Up Robust Features (ASURF), Vector of Locall Aggregated Descriptos (VLAD) y Cloud Forest (CF) en la extracción de características faciales. También, indica que el algoritmo CF ha mostrado mejoras en la clasificación de los conjuntos de datos faciales usados en el procesamiento de imágenes que están restringidas con traducción, rotación, escala, color, iluminación y distorsión.

En la investigación presentada por Zhang y Zhang (2018), plantea la solución de un algoritmo el cual permite realizar la identificación y seguimiento de un objetivo, de un rostro humano mediante video, el cual se basa en el algoritmo de seguimiento Camshift y el algoritmo de seguimiento KCF, evidenciando una comparación en ambos algoritmos.

***1.4.1.2.1 Internacionales*.** La definición de inteligencia artificial explicada a grandes rasgos qué es la capacidad de los ordenadores de brindar una respuesta rápida a las órdenes de un servidor y al mismo tiempo de aprender de los datos recibidos y poder generar su propia toma de decisiones, su aprendizaje es similar al de un humano pero con la diferencia de que la inteligencia artificial puede analizar varios resultados al mismo tiempo superando así la inteligencia de un humano la inteligencia artificial sirve para mejorar y beneficiar a las personas en la eficiencia en tiempo y calidad en la vida (Lasse-Rouhiainen, 2018).

Investigaciones acerca de ciudades inteligentes en donde han participado más de 200 ciudades de todo el mundo. El último estudio realizado se enfoca en necesidades como el cambio climático, la visión al futuro, los programas de apoyo al avance tecnológico y, muy importante, la seguridad ciudadana. Los países que tienen mayores avances son países asiáticos, destacando en el primer puesto Singapur (Smart City Governments, 2021).

Corea del sur con Seúl y ciudades como Londres, Barcelona y claro, Estados Unidos. Que ha tenido avances significativos como lo vemos en una empresa (Smart, Scalable and Intuitive Security Solutions, 2022) que posee una estructura de vigilancia mediante (VMS) que son sistemas de administración de videos usando diferentes métodos y entre ellos el uso de la inteligencia artificial.

1.4.1.2.2 Nacionales. El desafío de implementar sistemas que emplean inteligencia artificial se encuentra limitado a la poca inversión monetaria e investigativa como en Colombia. Como resultado, se usan sistemas convencionales, con poca calidad de imagen, bajo ángulo de visión y sin un sistema que permita almacenar las características faciales de los individuos, que permitan la identificación, individualización y judicialización.

En Colombia se usa la inteligencia artificial para la detección de placas de vehículos o por ejemplo el Sistema Inteligente De Monitoreo Integral Móvil (SIMIM), que cuenta con cámaras inteligentes y que fue instalado en la ciudad de Medellín con el fin de monitorear la ciudad y cuenta con tecnología para reconocimiento facial y de placas que permitieron la disminución de hurtos en dicha ciudad (Policía Nacional de Colombia-POLNAL, 2022).

Hay sistemas de seguridad vial que permiten a las entidades de control tener acceso a datos vehiculares tales como la placa, el color, la marca y hasta la velocidad del auto en cuestión y esto gracias a un sistema de cámaras que mediante dispositivos IoT permiten el análisis y procesamiento de los datos obtenidos en las cámaras. De esto se encarga la entidad llamada Centro De Comando, Control, Comunicaciones Y Cómputo (C4) (POLNAL, 2022) Este centro contaba en el año 2015 con alrededor de 267 cámaras y en la actualidad cuenta con más de 3.000 en la capital y más de

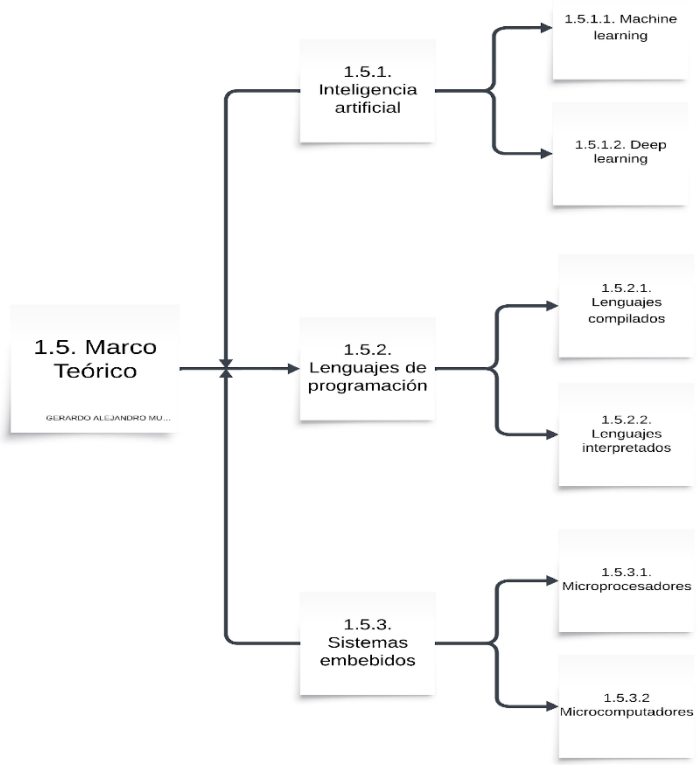
10.000 en todo Colombia, también cuenta con un helicóptero y 5 drones que ayudan a la vigilancia, control de operativos y monitoreo a la ciudad.

1.4.2 Marco teórico

A continuación, se describe cómo se aplican las fases en función de la taxonomía desarrollada para el trabajo de grado.

Figura 1.

Taxonomía



1.4.2.1 Inteligencia artificial. La labor principal de la inteligencia artificial es imitar la función cognitiva humana y esta a su vez realizar una labor de manera ordenada de tal manera que parezca una acción humana. Cómo reconocer gestos y reacciones en una situación cualquiera. La inteligencia artificial es una herramienta capaz de proporcionar reconocimiento para predicción. Actualmente, es capaz de recolectar datos, organizar, analizarlos y tomar decisiones propias para generar un resultado. Esto es importante ya que en su capacidad está inferir para desarrollar trabajos que antes eran solo reservados para los humanos (Garcí-Serrano, 2012) (Lasse-Rouhiainen, 2018).

La inteligencia artificial basada en redes neuronales hace uso del Machine Learning o aprendizaje automático. Es una rama de la IA, Que como su nombre indica, tiene la capacidad de aprender sobre patrones o datos que se repiten y esto de forma autónoma mediante una serie de capas neuronales la cual se subdivide en supervisada o no supervisada.

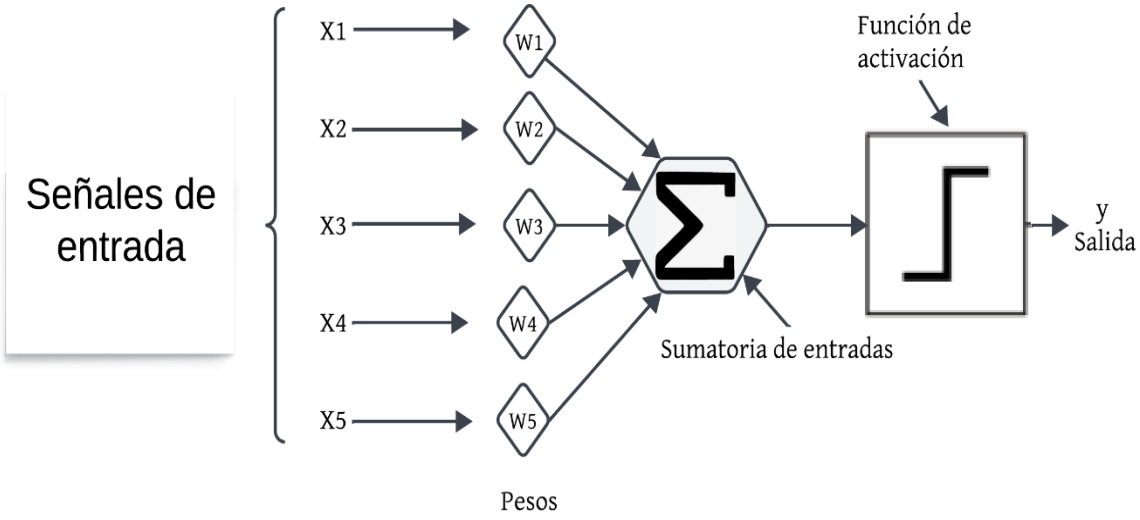
1.4.2.1.1 Machine learning. El aprendizaje automático, es un subcampo de la inteligencia artificial y la ingeniería electrónica, se centra en desarrollar algoritmos que permiten a las computadoras aprender de los datos para mejorar en tareas específicas. Estos algoritmos, basados en estadísticas, reconocimiento de patrones y optimización, identifican patrones y toman decisiones automáticamente. Se aplica en áreas como visión por computadora y procesamiento de lenguaje natural, ofreciendo soluciones automatizadas para problemas complejos (Manrique-Rojas, 2020).

La clasificación del aprendizaje automático se da por su tipo de aprendizaje. El aprendizaje automático supervisado trabaja con datos etiquetados y se usa para aprender iterativamente sobre datos de entrada y generar predicciones sobre la salida de estos; teniendo como entrada estructuras de cualquier tipo de información que pasan por este procesamiento y luego generar modelos que predicen o clasifican estos datos de salida. A diferencia del no supervisado que no necesita datos etiquetados para entrenar. Es más usado como algo exploratorio sobre una información que no necesita un dato de salida, si no una agrupación de estos datos y una respectiva consideración. También se tiene el aprendizaje reforzado, el algoritmo aprende a tomar decisiones observando su entorno. A través de un proceso de prueba y error, el algoritmo recibe recompensas o penalizaciones basadas en las acciones que realiza, y se entrena sin la necesidad de datos clasificados o no clasificados (Pertuz, C. M. P. , 2022).

En el campo del aprendizaje automático, una de las herramientas más significativas es la red neuronal artificial, que se inspira en el funcionamiento del cerebro humano (Tahir, ul-Hassan, & Asghar Saqib, 2016). La Figura 2 ilustra un modelo de una red neuronal, que es un componente esencial en la implementación de algoritmos de aprendizaje supervisado y no supervisado.

Figura 2.

Modelo matemático básico de redes neuronales



Se observa que la entrada Xn se somete a una suma ponderada, donde cada conexión tiene un peso específico Wn. Luego, para introducir no linealidades y capturar relaciones más complejas en los datos, se aplica una función de activación. Esta función transforma la suma ponderada en una salida no lineal, permitiendo a la neurona aprender patrones más preferidos y hacer que la red sea más adaptable a datos complejos y no lineales se describen en las ecuaciones:

Ecuación 1.

Salida de la neurona

La ecuación 1 representa la salida de la neurona, que es calculada por la función de activación aplicada a la suma ponderada de las entradas:

$$Y(X) = \sigma(U) \qquad (1)$$

Donde Y(X) es la salida de la neurona dada la entrada X y es la función de activación. Siendo σ la función de activación que se define de la siguiente forma.

Ecuación 2.

Función de activación

En la ecuación 2 se define la función de activación, que en este caso es una función escalón. La función de activación determina si la neurona se activa o no, dependiendo del valor de la suma ponderada *U*:

$$\sigma = \{1 \quad si\ U > 0\ 0 \quad si\ U\ \leq 0 \qquad (2)$$

Esta función de activación es crucial ya que permite a la red introducir no linealidades, lo que es esencial para modelar relaciones complejas en los datos. Y por último (U) sería:

Ecuación 3.

Cálculo de la entrada para función de activación

La Ecuación 3 describe cómo se calcula la entrada neta a la neurona antes de aplicar la función de activación:

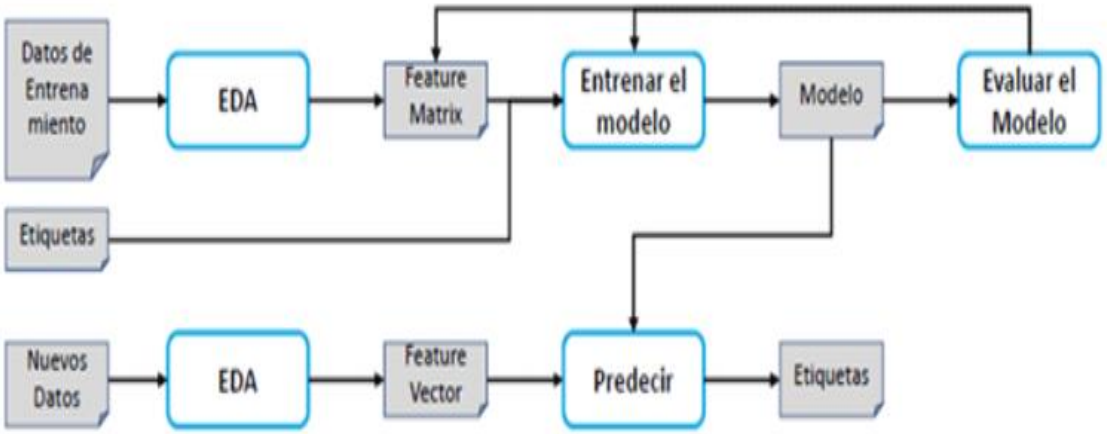
$$U = X1 + W1 + X2 + W2 + X3 + W3 \ldots . - \theta \qquad (3)$$

Y siendo θ el valor de e l valor umbral el cual es el modelo llamado **Lógica Umbral**.

Tras comprender el modelo matemático fundamental de una red neuronal, es esencial explorar cómo se aplica este modelo en un flujo de trabajo de aprendizaje automático. La Figura 3, describe el proceso completo desde la preparación de los datos hasta la realización de predicciones.

Figura 3.

Proceso de aprendizaje automático supervisado

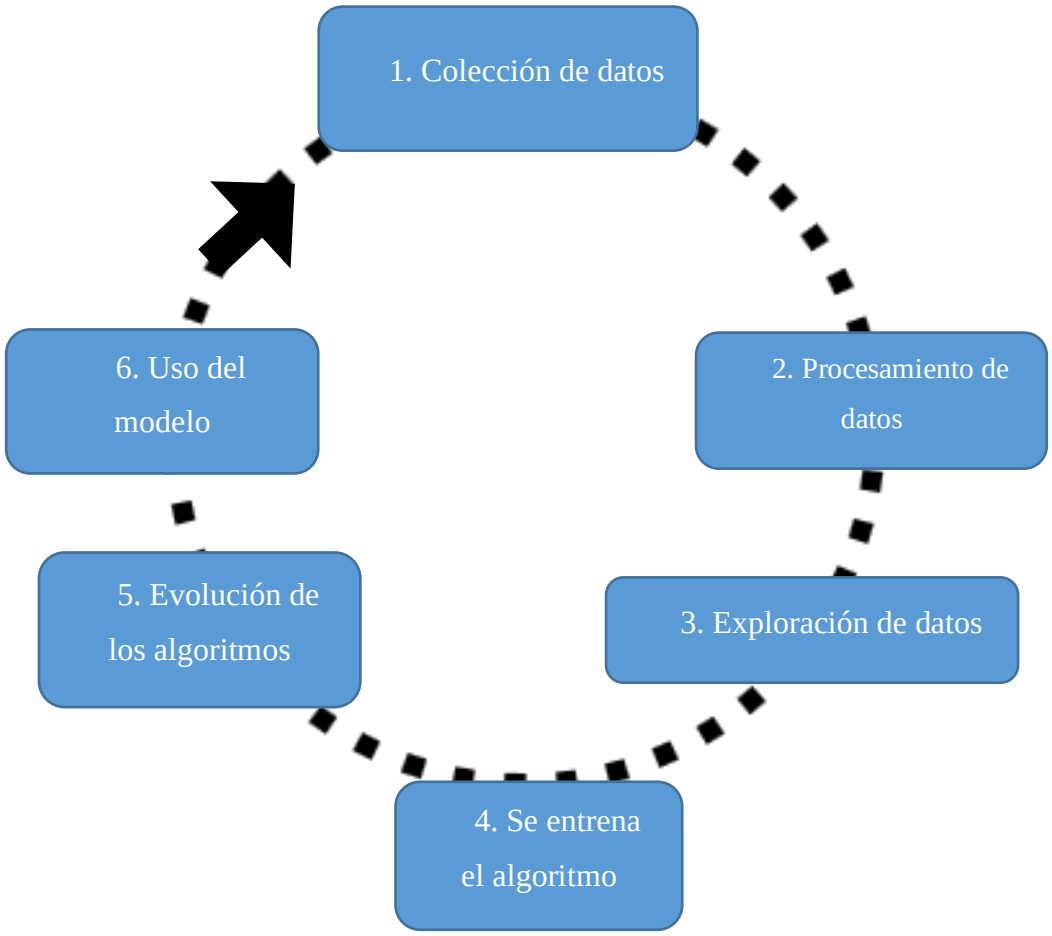


Nota. Fuente: Máster en Data Science, n. d.

La conceptualización de la red neuronal y el procesamiento de datos dentro de ella nos conducen naturalmente al siguiente paso de nuestra exploración: la construcción de un modelo de Machine Learning. La Figura 4 ilustra los pasos secuenciales involucrados en este proceso.

Figura 4.

Pasos para construir un modelo de machine Learning



Rojas, E. M. (2020), describe cada uno de estos pasos como se describe a continuación.

Explicación por etapas.

Colección de datos: Inicialmente se reúne un conjunto de bases de datos imágenes de dos clases de rostros, las cuales previamente han sido etiquetadas 0,1 con la identidad correspondiente a *"vanessa"* y *"Alejandro"* respectivamente lo que permite el entrenamiento supervisado del modelo de clasificación. La base de datos se ha dividido utilizando un esquema de validación del 80/20 lo

que significa que el 80% de las imágenes se utilizarán para entrenar el modelo y el 20% restante de las imágenes se reserva para validar el modelo.

Procesamiento de datos. En esta etapa se preparan la base de datos en donde las imágenes de los rostros deben estar en un formato estándar antes de que puedan ser procesadas por un modelo de machine learning

Exploración de datos. En esta etapa se observan en la base de datos las dos clases de rostros y una cierta diferencia en luminosidad.

Se entrena el algoritmo. Durante el proceso de aprendizaje automático supervisado, se lleva a cabo el entrenamiento del modelo, etapa en la que se aplica un algoritmo de aprendizaje automático para aprender de los datos proporcionados. Para este proyecto de grado se empleó YoloV8, VGG y ResNet50. Con YoloV8, se ajustó meticulosamente los hiper parámetros para optimizar el rendimiento del modelo. Para el modelo de VGG. Se usó la versión VGG16 con la misma base de datos, pero sin requerimiento de labels o etiquetas previas. Es decir que al modelo se adjuntan las imágenes requeridas y este se entrena teniendo en cuenta las carpetas por cada clase. Para el modelo de ResNet50 se usaron los mismos parámetros.

Evaluación de los algoritmos. Durante esta fase se realiza una revisión y un refinamiento de sus hiperparametros con el fin de llegar a un entrenamiento más óptimo de los modelos ya entrenados.

Uso del modelo. En esta etapa final se integra el modelo ya entrenado en un dispositivo de hardware específico, como una NVIDIA Jetson Nano, la cual es diseñada específicamente para aplicaciones de inteligencia artificial y aprendizaje automático en el borde (Edge computing).

1.4.2.1.2 Deep learning Deep Learning también conocido como aprendizaje profundo, es una rama de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos computacionales inspirados en la estructura y funcionamiento del cerebro humano, específicamente en las redes neuronales artificiales. Estas redes neuronales profundas están compuestas por múltiples capas de nodos, que permiten el procesamiento de datos de manera jerárquica y la extracción de características cada vez más abstractas (J. Schmidhuber. 2015).

El objetivo principal del *Deep Learning* es capacitar a los modelos para que puedan aprender a partir de grandes volúmenes de datos, sin necesidad de una programación manual específica para cada tarea. Esto se logra mediante el uso de algoritmos de aprendizaje supervisado o no supervisado, en los cuales se ajustan los pesos y las conexiones entre las neuronas para que el modelo pueda realizar tareas como reconocimiento de imágenes, reconocimiento de voz, traducción automática, procesamiento del lenguaje natural, entre otras. También ha demostrado ser muy eficaz en diversas áreas, ya que su capacidad para manejar grandes cantidades de datos y aprender representaciones complejas ha llevado a avances significativos en el campo de la visión por computadora, la medicina, la robótica, entre otros. Permite abordar problemas difíciles y obtener resultados precisos. El *Deep Learning* se utiliza para entrenar modelos de aprendizaje automático que pueden aprender y realizar tareas complejas a partir de datos brutos sin la necesidad de una programación detallada (J. Schmidhuber. 2015).

1.4.2.1.3 Lenguajes de programación. Existen diferentes lenguajes de programación que se usan para implementar la inteligencia artificial. Los lenguajes de programación permiten escribir una serie de instrucciones en forma de algoritmo que permiten controlar un comportamiento lógico físico de un sistema Hardware/Software.

Los lenguajes de programación se dividen en tres grupos y se clasifican en lenguajes de alto nivel, de medio nivel y de bajo nivel (Gortázar-Bellas et al., 2016). Y para eso se usan compiladores y traductores que son más fáciles de usar, estas adaptaciones que hace el lenguaje con un compilador permiten la comunicación más rápida y eficaz entre el hombre y la máquina y se realizan en programas llamados IDE de programación. Ahora, los lenguajes de programación empleados en inteligencia artificial son Python y C + +.

Lenguaje Compilado. Este requiere una serie de pasos antes de poder ejecutarlo. Se debe compilar para poder pasar a lenguaje máquina y en ese momento poder correr el programa. Es decir, toma todo el código fuente y en un proceso lo convierte en lenguaje máquina el cual ya luego pasa a ser ejecutado y a cumplir con las órdenes, algunos ejemplos son:

C.

C++

Objective-C

Lenguaje Interpretado. Este lenguaje al contrario del anteriormente mencionado necesita ejecutar cada línea a lenguaje máquina. Es decir. Cada vez que se lee una línea de código, este se procesa y se convierte a lenguaje máquina. Y este se ejecuta directamente en Matlab, R, Python o Ruby.

En la implementación de la Inteligencia artificial se usan los lenguajes de programación de alto nivel, ya que estos son capaces de ejecutar programas complejos con un tiempo de ejecución que es crucial para el procesamiento del código fuente.

Teniendo como base esto y lo analizado anteriormente en la investigación. Python es el lenguaje más apto, dinámico y versátil para la programación de inteligencia artificial, ya que es multiplataforma y no es necesario compilar para cada plataforma Y es compatible con distintas bibliotecas de IA. Este mismo necesita paquetes y librerías que le permitan sacar un mayor provecho. Estas herramientas vienen dadas a su capacidad y aplicación. Dado que hay miles de aplicaciones y muy específicas en los diferentes códigos y herramientas para el aprendizaje automático. A continuación, se presenta en la tabla 3 las librerías más usadas con Python

Python es un lenguaje de programación interpretado, de código abierto, de alto nivel multiplataforma y que está orientado a objetos, programaciones imperativas y también en menores

medidas, programación funcional y que se usa para desarrollar aplicaciones en informática y desarrollo web (Centro De Información Para La Industria, 2022). Python es compatible con distintas bibliotecas como se puede observar en la Tabla 3.

Tabla 3.

Características de las diferentes librerías

Librería	Característica	URL
OpenCV	Es una librería libre. Que permite ser usada para investigación o comercial. Es multiplataforma. Tiene la capacidad de comprender ángulos del rostro y estimar la pose para predecir. Y no solo para rostro sino también para objetos.	(Raúl Igual & Carlos Medrano, 2008)
Keras	En una librería de redes neuronales escrita originalmente por Python. Analiza redes de aprendizaje profundo siendo versátil con el tiempo. Posee optimizadores matemáticos dados por desarrollo de tensorFlow. Modelos profundos de TPU Y GPU	(Antona Cortés & Carlos, 2007)
Tensor Flow	Librería de código abierto para aprendizaje automático por medio de tareas Construye y entrena redes neuronales para descifrar patrones y correlaciones. Crea estructura de datos que describen el movimiento de los mismos en una serie de nodos de procesamiento	(Valle Barrio & Alfredo, 2018)
TFX	Forma parte de una extensión de tensorFlow para redes neuronales, pero este va en una forma escalable y por medio de capas.	

Pandas	Es una extensión de Numpy que sirve para manipulación y análisis de datos series temporales y tablas numéricas	("Introducción a La Librería Pandas," 2022)
Numpy	Es una librería que da soporte matemático, con vectores, matrices. Grandes y multidimensionales como por ejemplo ndarray	(Alfredo Sánchez Alberca, 2016b)
SHAP	Sirve para predecir variables usando machine learning usando cálculos de la teoría de juego Pero usa mucha memoria lo que lo hace lento	(Scott Lundberg, 2018)
Matplotlib	Es una librería que sirve para generar gráficos que van de la mano con Numpy por ejemplo.	(Alfredo Sánchez Alberca, 2016a)
Scikit-learn	Es una librería de aprendizaje automático que sirve para algoritmos de clasificación, agrupamiento, aumento de gradientes y regresión. Y se basas también con librerías como Numpy y Scipy	(UNIVERSIDAD DE ALCALÁ, 2022)
Scipy	Tiene como función la informática científica y técnica de Python. Se dice que es el núcleo de las capacidades científicas y está llena de subpaquetes que complementan al resto de librerías. Por ejemplo. Integrar, interpolar, espacial, constantes, clúster. Etc.	(Jorge A. Pérez Prieto, 2019)

1.4.2.**2 Edge computing**. Para la aplicación en un sistema embebido se usará *Edge Computing* que es un paradigma informático que se refiere al procesamiento y almacenamiento de datos cerca del punto de origen o consumo de esos datos, en lugar de enviarlos a través de una red hacia un centro de datos remoto o la nube. En el contexto del reconocimiento facial, el *Edge Computing* se utiliza para realizar el procesamiento y análisis de datos relacionados con la detección y reconocimiento de rostros en dispositivos o sistemas ubicados en el borde de la red, como cámaras de vigilancia, dispositivos móviles o sensores.

Ventajas que tiene utilizar la plataforma de Edge computing para el reconocimiento facial son la latencia reducida ya que el procesamiento de datos en el borde de la red reduce la latencia al evitar la transmisión de datos a través de la red hacia un centro de datos remoto. Y la eficiencia de ancho de banda porque el procesamiento en el borde reduce la carga en la red y el uso de ancho de banda al realizar tareas de procesamiento localmente, evitando transmitir grandes volúmenes de datos a través de la red. En la Tabla 4 presenta los sistemas que son compatibles con *Edge computing.*

Tabla 4.

Tabla comparativa de sistemas embebidos compatibles con Edge Computing

Tarjeta	**Procesador /Chip**	**GPU**	**Acelerador de IA**	**Memoria RAM**	**Aplicaciones Comunes**	**Bibliotecas/Lenguajes Compatibles**
NVIDIA Jetson Nano	Quad-core ARM Cortex-A57	128-core Maxwell	Tensor Cores	4 GB LPDDR4	Visión artificial, robots	Tensor Flow, PyTorch, C++, Python
NVIDIA Jetson Xavier NX	Hexa-core NVIDIA Carmel ARM v8.2 64-bit	384-core Volta	Tensor Cores	8 GB LPDDR4x	Robótica avanzada, drones	Tensor Flow, PyTorch, C++, Python
NVIDIA Jetson AGX Orin	12-core ARM Cortex-A78AE	2048-core Ampere	Tensor Cores	32 GB LPDDR5	Autonomía, robótica compleja	Tensor Flow, PyTorch, C++, Python
Google Coral Dev Board	Quad-core ARM Cortex-A53	None (Edge TPU)	Edge TPU	1 GB LPDDR4	Dispositivos IoT, detección de objetos	Tensor Flow Lite, C++, Python

Google Coral USB Accelerator	Edge TPU	None (Edge TPU)	Edge TPU	N/A	Inferencia IA en dispositivos existentes	Tensor Flow Lite, C++, Python
Raspberry Pi 4	Quad-core ARM Cortex-A72	Broadcom VideoCore VI	No (Compatible con Coral USB)	Hasta 8 GB LPDDR4	Prototipos IA, computación general	Tensor Flow Lite, C++, Python
Intel NCS2	Intel Movidius Myriad X VPU	None	VPU	N/A	Visión por computadora, IA ligera	Open VINO, C++, Python
Intel Movidius Myriad X	Intel Movidius Myriad X VPU	None	VPU	N/A	Visión por computadora, IA ligera	Open VINO, C++, Python
Qualcomm Snapdragon 845/865	Kryo 385/485	Adreno 630/640	DSP Hexagon 685/690	4 GB LPDDR4	Dispositivos móviles, AR/VR	TensorFlow, C++, Python
Qualcomm QCS610	Hexa-core ARM Cortex-A53	None	DSP	2 GB LPDDR3	Dispositivos IoT, visión artificial	Tensor Flow Lite, C++, Python
Hailo-8	Hailo-8 Processor	None	Hailo-8	N/A	Visión por computadora, IA en tiempo real	TensorFlow, C++, Python

Xilinx Kria KV260	Quad-core ARM Cortex-A53	Mali-400MP2	No (Compatible con aceleradores externos)	4 GB LPDDR4	Visión artificial, procesamiento de señales	Tensor Flow, Vitis AI, C++, Python

Dadas las características técnicas mostradas en Tabla 4, la serie de NVIDIA Jetson es la opción más indicada. Debido a que ofrece un equilibrio entre potencia de cómputo (núcleos Volta y Tensor Cores), capacidad de ejecutar modelos complejos de aprendizaje profundo en tiempo real, y compatibilidad con bibliotecas como TensorFlow y PyTorch. Permitiendo procesar tareas de visión artificial, como el reconocimiento facial, con precisión y eficiencia energética, lo que es necesario para aplicaciones biométricas en entornos de Edge Computing.

.

1.5 Metodología

En la Tabla 5 se presentan un análisis de los modelos más influyentes en cuanto clasificación y detección en objetos y reconocimiento facial cada uno con sus diferentes arquitecturas entre ellos se encuentran YoloV8 y su versión V5, LBP (*Local Binary Pattern*), HAAR, ResNet50Y VGG.

Tabla 5.

Análisis de modelos

Modelo	Precisión	Técnica de detención	Velocidad de procesamiento	Entrenamiento
YOLO V8	Muy alta	Redes Neuronales Convolucionales profundas para detección en tiempo real,	Muy rápida	Requiere gran cantidad de datos y recursos computacionales
LBP	baja	Uso de patrones binarios locales para reconocimiento de características en imágenes.	Moderada	Requiere menos datos y es menos intensivo en recursos
VGG	Alta	Red profunda	Moderada	Requiere gran cantidad de datos, recursos computacionales, además de un gran tiempo de entrenamiento
Haar	Moderada	Análisis de características de Haar para identificación de rasgos faciales	Rápida	Necesita un conjunto de datos Moderado y ajustes manuales
ResNet50	Alta	Red neuronal convolucional (CNN)	Rápido	Requiere una cantidad significativa de recursos computacionales debido a su

				arquitectura profunda y la gran cantidad de parámetros que debe aprender durante el entrenamiento
YOLO V5	no es el modelo más preciso en términos absolutos	- FCOS (Fully Convolutional One-Stage) p	La velocidad de procesamiento varía según el tamaño del modelo	YOLOv5 puede entrenarse con una variedad de tamaños de datos,

A lo largo de las distintas fases del proyecto, se ha seguido un enfoque estructurado y metódico, dividido en varias fases clave. Cada etapa ha sido abordada meticulosamente, desde la selección de sistemas y algoritmos hasta la implementación y evaluación de rendimiento en entornos de *Edge Computing*. El proyecto se desarrollará en tres fases claves, presentadas a continuación:

1.5.1 Fase 1

Base de datos y criterios de búsqueda. Con la finalidad de lograr los resultados deseados, se presenta una metodología de mapeo sistemático, que permite identificar y comprender los lineamientos para abordar el desarrollo del proyecto.

Análisis comparativo. Investigar y evaluar diferentes sistemas de autenticación biométrica basados en reconocimiento facial disponibles en el mercado. escogiendo los óptimos y comparando sus características, precisión, velocidad.

Selección. Seleccionar el sistema embebido que mejor se adapte a las necesidades del proyecto se ajuste a los criterios de precisión, velocidad y seguridad, asociados con la implementación y el mantenimiento del sistema. Finalizando la Fase 1, caracterización del sistema para un método

computacional y la aplicación de inteligencia artificial mejorando así el proceso de reconocimiento de rostros.

1.5.2 Fase 2

La investigación se centra en la evaluación y comparación de algoritmos de identificación y clasificación de características faciales, seleccionados previamente en la Fase I por su potencial. El objetivo es analizar meticulosamente el rendimiento de estos algoritmos en condiciones de iluminación y poses variables, utilizando métricas clave para determinar su eficacia. Este análisis nos permitirá identificar los algoritmos más robustos y precisos, adecuados para entornos reales y desafiantes en el reconocimiento facial. Estos factores han sido identificados como elementos críticos que afectan la eficacia en la detección de rostros, y su mejora es esencial para avanzar en la precisión y confiabilidad de los sistemas de autenticación biométrica.

Implementación y entrenamiento. Para la implementación y entrenamiento de los modelos que resultaron de la primera fase, se usará los modelos Yolo en su versión V8 (Ultralytics YOLOv8) y su versión V5, VGG (Visual Geometry Group), ResNet50 (Red Residual de 50 capas), HAAR, LBP y usando el modo de entrenamiento el cual implica proporcionar una base de datos y ajustar sus parámetros para lograr que pueda realizar predicciones precisas.

Para crear la base de datos se emplearon 893 imágenes que fueron capturadas por la cámara de un dispositivo móvil, dichas imágenes se dividieron en dos etiquetas correspondientes a dos clases "Vanessa" Y "Alejandro". Para eso se usa el 73% de las imágenes para entrenamiento y el 26% para validación. Para las versiones v5 y v8 de Yolo fue necesario dimensionar las imágenes a 640*640 píxeles, para el modelo de HAAR y LBP se dimensionan a 24*24 píxeles y para VGG y Resnet50 se dimensionan en 224*224 a este conjunto de datos se le aplicó la técnica de aumento de datos (*Data Augmntation*) con el fin de obtener un entrenamiento más óptimo.

Evaluación de rendimiento. Durante la evaluación de los seis modelos, se consideraron tanto las métricas de rendimiento como el tiempo de ejecución de cada uno, y la versión óptima en términos de tiempo de ejecución de cada modelo, con el objetivo de identificar los más eficientes,

tal como se muestra en la Tabla 6. Para evaluar los modelos entrenados, se emplearon métricas claves como precisión, *Recall* y *Accuracy*, las cuales son fundamentales para determinar cuál de los algoritmos ofrece la detección y reconocimiento de rostros. Además, emplea el optimizador *Adam* a los modelos para ajustar los pesos durante el entrenamiento, para mejorar la precisión general.

Tabla 6.

Tiempo de ejecución

Modelo	**Tiempo de ejecución**
VGG	3,20h
YOLOV8	0,229h
RESNET50	2,30h
YOLO V5	0,540h
HAAR	9,34h
LBP	10,45h

Teniendo en cuenta las especificaciones anteriores, se seleccionaron los tres modelos que cumplían con los requisitos: VGG, YOLOv8 y ResNet50. Estos modelos fueron sometidos a la etapa de entrenamiento bajo las mismas condiciones, utilizando 60 épocas, el optimizador "Adam", la misma base de datos y se usó la función *Early Stopping* la cual detiene el entrenamiento cuando el rendimiento en un conjunto de validación comienza a empeorar, evitando así el sobreajuste. Como resultado, se obtuvieron las gráficas correspondientes de *Accuracy, Recall, Precision* y las matrices de confusión para cada modelo. Además, se analizaron los datos obtenidos en cada época y sus métricas correspondientes, con el objetivo de evaluar la mejora de los modelos entrenados. Este análisis del entrenamiento detallado es fundamental para desarrollar algoritmos de autenticación biométrica más robustos y eficaces.

1.5.3 Fase 3

Configuración de la plataforma de Edge Computing. Tras obtener resultados favorables, el entrenamiento se implementa en un sistema de Edge Computing, utilizando en la primera fase un kit de desarrollo Jetson Nano. Para configurarlo, se utiliza una tarjeta microSD con capacidades de lectura y escritura superiores a 100 Mb/s para garantizar una velocidad adecuada. Se descarga la imagen ISO de Jetpack 4.6 desde el sitio oficial de NVIDIA, y se siguen las instrucciones proporcionadas por la compañía para completar la configuración. Posteriormente, se instalan los requisitos necesarios para cada modelo. En este proceso, se emplea la herramienta Docker para crear contenedores, lo que evita conflictos entre versiones de software y facilita la ejecución de los modelos.

Análisis de resultados. Las pruebas finales son cruciales para validar el rendimiento del dispositivo en condiciones reales y garantizar que cualquier inconveniente sea resuelto antes de su implementación en un entorno práctico. Finalmente, se verifica la funcionalidad de los modelos usando el entorno de desarrollo Visual Studio Code, usando una versión de Python 3.10 y cargando los pesos entrenados que se guardan en un formato .pt (Pytorch) en la plataforma Jetson Nano, con el objetivo de visualizar en tiempo real la correcta detección de las clases. Estos resultados son fundamentales para evaluar la efectividad de una herramienta de control de acceso doméstico basada en inteligencia artificial y aprendizaje automático supervisado.

1.5.4 Diseño Metodológico

1.5.4.1 Línea de investigación. La presente investigación se enmarca en la línea de investigación "Desarrollo Mecatrónico" y área temática de Ingeniería Computacional del programa de Ingeniería Mecatrónica y el grupo GRIM, debido a que el desarrollo involucra la implementación de algoritmos y técnicas computacionales en el campo de la autenticación biométrica basada en el reconocimiento facial y el aprendizaje automático en una plataforma de *Edge Computing*.

1.5.4.2 Tipo de investigación. El trabajo de grado se enmarca en una investigación aplicada, experimental y cuantitativa, la cual se centra en la utilización de conocimientos y saberes para la resolución práctica de problemas relacionados con el procesamiento de imágenes mediante algoritmos de aprendizaje automático. Se busca realizar una evaluación cuantitativa y comparativa de los algoritmos en el proceso de inferencia para la autenticación biométrica, en una plataforma de *Edge Computing*.

1.5.4.3 Hipótesis de la investigación. ¿El análisis comparativo de los algoritmos para el reconocimiento facial permite autenticar de manera eficaz el rostro de una persona para el monitoreo de un hogar?

1.5.4.4 Identificación y Gestión de Riesgos. En el desarrollo de este proyecto, se identifican varios riesgos técnicos que podrían afectar el sistema de autenticación biométrica basado en reconocimiento facial y Edge Computing. A continuación, se detallan los riesgos potenciales y las estrategias de mitigación para cada uno:

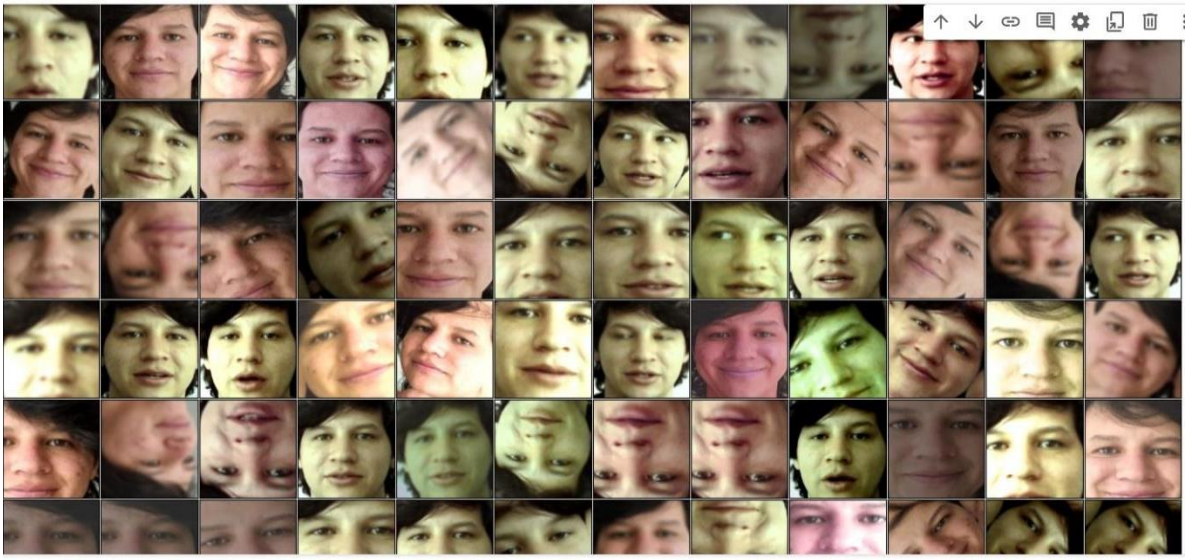
Errores en los Datos.

Riesgo. La presencia de errores en los datos de entrenamiento, como imágenes mal etiquetadas o de baja calidad, puede afectar significativamente el rendimiento de los modelos.

Mitigación. Se implementará un riguroso proceso de limpieza y verificación de los datos antes del entrenamiento, asegurando la integridad y calidad del conjunto de datos. Además, se utilizarán técnicas de aumento de datos (*Data Augmentation*) para mejorar la diversidad y la robustez del conjunto de entrenamiento como se aprecia en la Figura 5.

Figura 5.

Imágenes con aumento de datos (Data augmentation)



Inestabilidad en la Plataforma de Edge Computing.

Riesgo. La Jetson Nano, como plataforma de Edge Computing, podría experimentar inestabilidad debido a sobrecarga de procesos o limitaciones de hardware, lo que podría afectar la ejecución de los modelos en tiempo real. Se debe tener en cuenta la capacidad de almacenamiento de la memoria Micro SD, ya que la cantidad de datos que se manejan son demasiado pesados.

Mitigación. Se monitorea constantemente el uso de recursos y se optimizará el código para minimizar la carga computacional. En caso de sobrecarga, se evaluarán alternativas como la reducción de la complejidad del modelo o la implementación de técnicas de optimización como TensorRT o ONNX

Sobrecarga de Recursos.

Riesgo. La ejecución de múltiples modelos simultáneamente o la necesidad de procesar grandes volúmenes de datos en tiempo real podría resultar en una sobrecarga de recursos, afectando la latencia y el rendimiento del sistema.

Mitigación. Se realizarán pruebas de carga para identificar los límites de la plataforma y se utilizará Docker para aislar procesos y gestionar eficientemente los recursos disponibles.

***Plan de contingencia*.** En caso de que los modelos no alcancen los niveles de precisión esperados o se presenten problemas significativos, se implementará el siguiente plan de contingencia:

Revisión y Ajuste de Hiperparámetros. Se realizará una revisión exhaustiva de los hiperparámetros de los modelos, como la tasa de aprendizaje, el número de épocas, y las funciones de pérdida, para optimizar su rendimiento.

Exploración de Nuevos Algoritmos. Si los modelos actuales no son suficientes, se explorarán otros algoritmos de aprendizaje automático y redes neuronales que puedan ofrecer mejores resultados en las condiciones específicas de este proyecto.

Optimización del Flujo de Trabajo. Se evaluará la posibilidad de implementar técnicas de reducción de dimensionalidad, como PCA (Análisis de Componentes Principales), para mejorar el rendimiento de los modelos.

Evaluación Continua en Condiciones Reales. Los modelos serán sometidos a pruebas en condiciones reales, fuera del entorno controlado de laboratorio, para identificar posibles debilidades y mejorar su capacidad de adaptación a diferentes escenarios.

1.5.4.5 Plan de acción.

Tabla 7.

Plan de acción

Objetivo	Actividad	Entregable	Recursos
Identificar y analizar los criterios de autentificación biométrica y aprendizaje automático basado en reconocimiento de rostros.	Investigar y evaluar diferentes sistemas de autenticación biométrica basados en reconocimiento facial disponibles en el mercado. Compara sus características, precisión, velocidad.	Documentos de investigación	Computador Acceso a Internet Bases de datos bibliográficas
	Realizar pruebas de campo de los sistemas seleccionados para evaluar su desempeño en situaciones reales. Recopila datos sobre la precisión de la autenticación, la velocidad de respuesta y la robustez	Códigos y tabla de comparación	Computador Acceso a Internet
	Analizar los resultados de las pruebas y seleccionar el sistema de autenticación biométrica basado en reconocimiento facial que mejor se ajuste a los criterios de precisión, velocidad y	diferentes pruebas con los algoritmos encontrados	Computador Acceso a Internet

	seguridad, asociados con la implementación y el mantenimiento del sistema.		
Evaluar y comparar diferentes algoritmos de identificación, extracción y clasificación de características para la detección de rostros en video basado en métodos de aprendizaje automático supervisado	Implementar y entrenar diferentes algoritmos de detección de rostros en video.	documentos de investigación y algoritmos	Computador Acceso a Internet
	Evaluar el rendimiento de los algoritmos en diferentes condiciones de iluminación y oclusión facial.	comprobación de algoritmos y análisis comparativo	Computador Acceso a Internet
	Realizar pruebas de rendimiento en tiempo real y comparar la velocidad de detección.	Analiza los resultados para identificar el algoritmo que proporciona una detección rápida y efectiva de rostros en tiempo real	Computador, cámara, software
Realizar una evaluación estadística de los algoritmos en una plataforma de Edge Computing a través de un conjunto de datos para la	Configuración de la plataforma de Edge Computing	código con la comprobación de las mejoras con respecto a los códigos encontrados	Computador, cámara, software

identificación de rostros, considerando factores como la precisión, robustez y repetibilidad del sistema	Implementación y evaluación de los algoritmos seleccionados en la plataforma de Edge Computing con los resultados obtenidos realiza una evaluación estadística comparativa basada en los factores de precisión, robustez y repetibilidad.	Edge Computing y función del código en la tarjeta de programación	computador, software, JetsonNano
	Analiza los resultados obtenidos y realiza comparaciones entre los algoritmos evaluados.	funcionalidad del sistema con la unión de la placa de programación y los diferentes componentes del sistema de seguridad	computador, software, Jetson Nano

1.5.4.6 Validación interna. La validación se realiza mediante la documentación de 3 modelos que presentan las mejores características, y arroja los resultados más convenientes para el reconocimiento facial y la aplicación en el monitoreo de hogares. Los algoritmos presentados en la Tabla 10 indican sus diferentes características de evaluación y la técnica que emplea cada uno.

En el diagrama de flujo se emplea el modelo de Yolov8 para el entrenamiento del dataset y este a su vez arroja un archivo Pytorch (.pt) que refiere al modelo ya entrenado. Este modelo se introduce nuevamente en un sistema ya sea de cómputo o embebido para procesar la información y por medio de una cámara USB se indican las etiquetas descritas anteriormente y se visualiza la a la persona presente en la cámara.

Para los algoritmos de VGG se obtiene un archivo Pytorch (.pt). el cual también se procesa en Python para su lectura y procesamiento usando una vez más un sistema y una cámara para observar la detección se usarán las siguientes métricas.

Para el modelo de ResNet50 se usa nuevamente la misma base de datos y se procede al entrenamiento. al finalizar este también guarda el modelo en un archivo Pytorch para su posterior lectura por medio de una cámara USB.

1.5.4.6.1 Métricas de evaluación para medir la precisión en un rostro de manera correcta. Para validar la exactitud, sensibilidad, especificidad, PPV (valores predictivos positivos), NPV (valores predictivos negativos) de los algoritmos se utilizan ciertas ecuaciones prestadas a continuación.

Ecuaciones.

Exactitud (Accuracy).

Ecuación 4.

Cálculo de confianza

(4)

$$Accuracy = \frac{(VP + VN)}{VP + FP + FN + VN}$$

Sensibilidad (Sensitivity).

Ecuación 5.

Cálculo de Recall

(5)

$$SENSITIVE = \frac{VP}{VP + FN}$$

Especificidad (Specificity).

Ecuación 6.

Cálculo de precisión

(6)

$$Specificity = \frac{VN}{VN + FP}$$

Valor Predictivo Positivo (Positive Predictive Value).

Ecuación 7.

Cálculo de verdaderos negativos (VN) (7)

$$PPV = \frac{VP}{VP + FP}$$

Valor Predictivo Negativo (Negative Predictive Value).

Ecuación 8. (8)

Cálculo de verdaderos positivos (Vp)

$$NPV = \frac{VN}{VN + FN}$$

Ecuación 9. (9)

Cálculo de falsos negativos (FN)

$$FN = \frac{FN}{VP + FN}$$

(10)

Ecuación 10.

Cálculo de falsos positivos (FP)

$$FP = \frac{FP}{FP + VN}$$

Ecuación 11. (11)

Cálculo de matriz de confusión

$$M = \begin{pmatrix} VP & FN \\ FP & VN \end{pmatrix}$$

Ecuación 12. (12)

Cálculo de F1 – SCORE

$$F1 = \frac{2 * precisión * recall}{precision + recall}$$

En estas Ecuaciones, los Verdaderos Positivos son los casos en los que el sistema identifica correctamente una cara real, los Verdaderos Negativos son los casos en los que el sistema identifica una cara falsa; los Falsos Positivos son los casos en los que el sistema identificó incorrectamente una cara falsa como real, y los Falsos Negativos son los casos en los que el sistema identificó incorrectamente una cara real como falsa.

De las métricas anteriormente mencionadas se usarán 4 que son esencial para la medición de la precisión de entrenamiento del modelo, que serán. Precisión, Recall, Confianza y matriz de confusión.

1.5.4.7 Validación externa. A continuación, se presenta en la Tabla 8 un análisis comparativo de los algoritmos más utilizados para el reconocimiento facial clasificándolos por su precisión, Recall, Accuracy, técnica utilizada en cada uno de ellos y sus respectivas ventajas.

La Tabla 8 resulta de la investigación, comparación y validación de los mejores algoritmos para la detección de rostros, clasificación y reconocimiento facial. Con los cuales se hará la validación externa de un algoritmo más completo usando la plataforma Edge Computing y el sistema embebido de Jetson nano de la mano con una cámara que usa una mejor resolución de 8 Mp varifocal de 5.5mm.

Tabla 8.

Tabla comparativa de algoritmos

Algoritmo	Precisión	Técnica de detención	Velocidad de procesamiento	Entrenamiento
YOLO V8	Muy alta (detecta con precisión en diversas condiciones de iluminación y orientación)	Redes Neuronales Convolucionales profundas para detección en tiempo real, lo que permite procesar imágenes rápidamente y con alta precisión.	Muy rápida (capaz de procesar imágenes en tiempo real)	Requiere gran cantidad de datos y recursos computacionales
VGG	Alta (VGG debido a su arquitectura profunda y el uso de redes convolucionales	Es una red profunda utilizada principalmente para la Clasificación de imágenes,	Moderada (su arquitectura profunda y su gran cantidad de parámetros	Requiere gran cantidad de datos, recursos computacionales, además de un gran

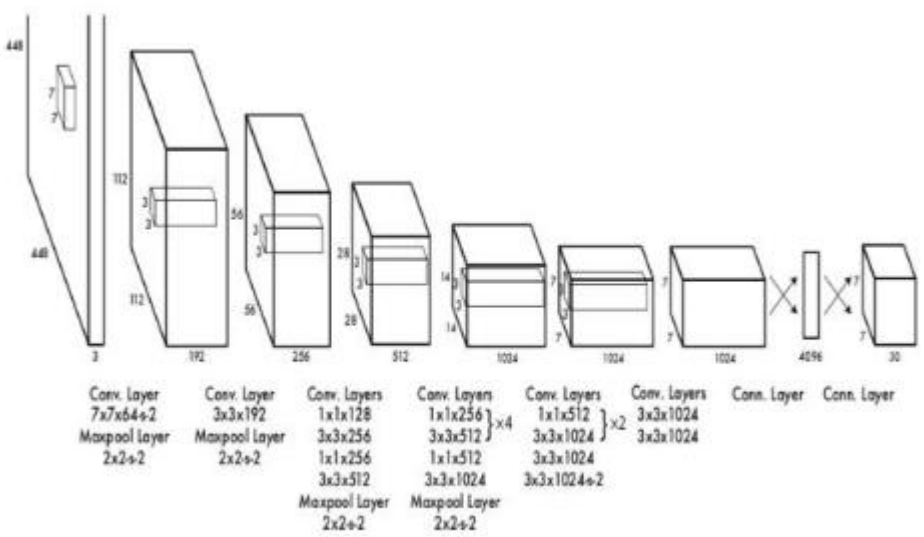
	con muchas capas)	conocido por aprender las características más detalladas	hacen que el procesamiento sea más lento)	tiempo de entrenamiento
ResNet50	Alta. Arquitectura de 50 capas, distribuidas en diferentes procesos del entrenamiento	red neuronal profunda diseñada para tareas de reconocimiento de imágenes	Moderada (gran cantidad de datos para cada capa. Conexión residual que hace que salte algunas capas)	Requiere gran cantidad de datos y recursos computacionales

A continuación, se van a mostrar las arquitecturas de cada modelo

Arquitectura de YOLO en la Figura 6. La red de detección tiene 24 capas convolucionales seguidas de 2 capas completamente conectadas. La alternancia de capas convolucionales 1×1 reduce el espacio de características de las capas anteriores. Se pre-entrenan las capas convolucionales en la tarea de clasificación de ImageNet. a la mitad de la resolución (imágen de entrada 224×224) y luego se duplica la resolución para la detección (Rozada Raneros, S. 2021).

Figura 6

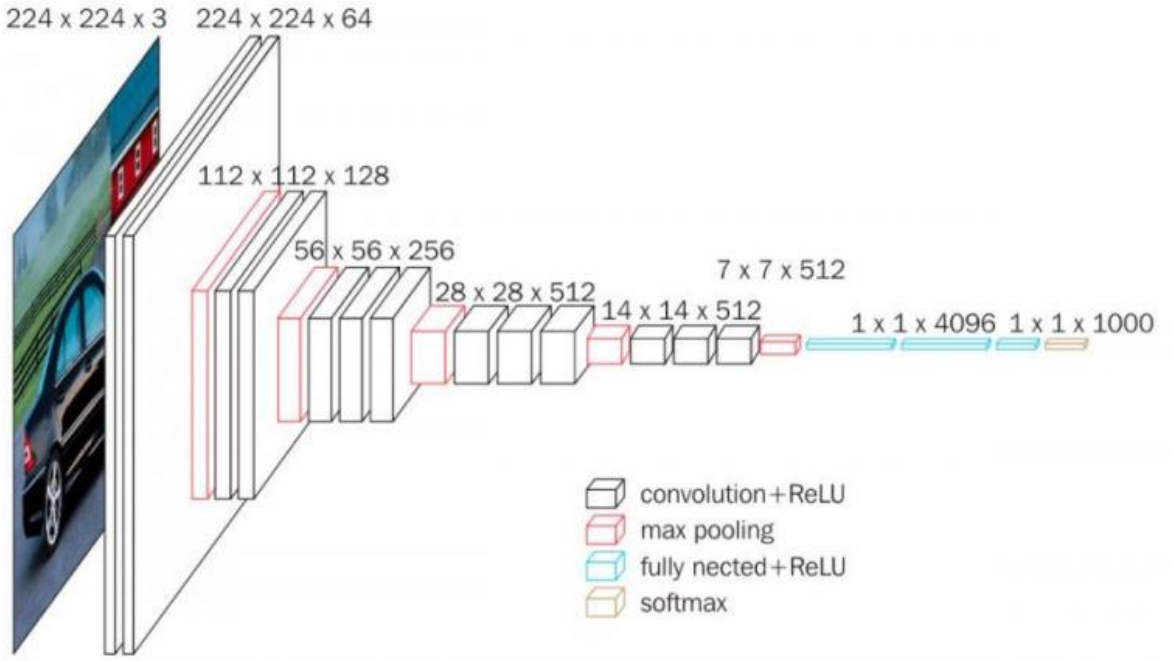
Arquitectura Yolov8



Arquitectura VGG que se muestra en la Figura 7, propuesta por (Simonyan & Zisserman, 2014), se trata de una arquitectura bastante simple, usando solo bloques compuestos por un número progresivo de capas convolucionales con filtros de tamaño 3×3. Además, para reducir el tamaño de los mapas de activación que se van obteniendo, se intercalan bloques *maxpooling* entre los convolucionales, reduciendo a la mitad el tamaño de estos mapas de activación. Finalmente, se utiliza un bloque de clasificación compuesto por dos capas densas de 4096 neuronas cada una, y una última capa, que es la de salida, de 1000 neuronas.

Figura 7.

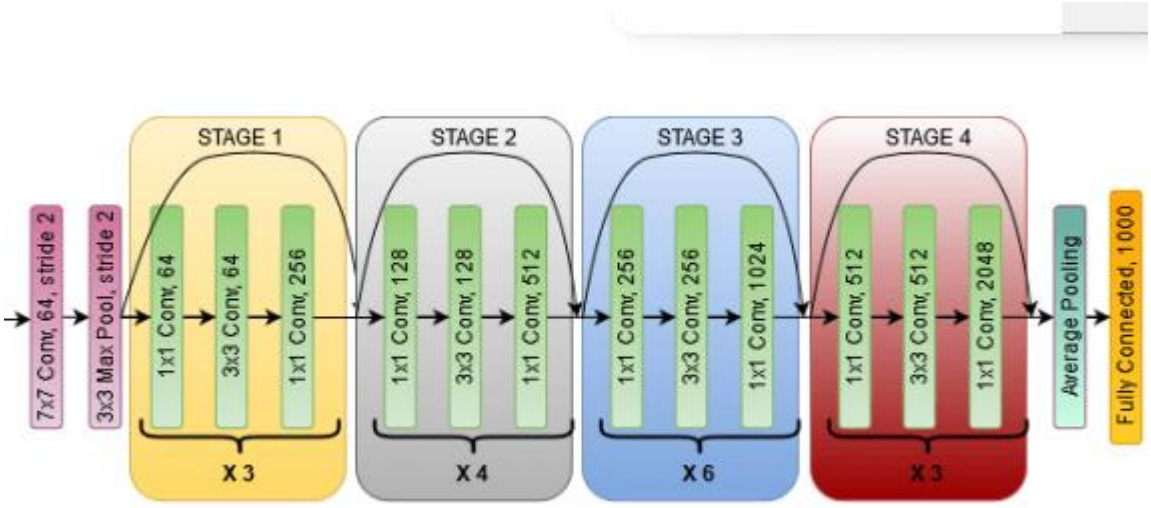
Arquitectura VGG



Arquitectura ResNet50 es una Red Neuronal Residual con 50 capas, su estructura es simple y efectiva, es beneficiosa por su amplia aplicación (K. He, X 2015) como se observa en la Figura 8. La red contiene un bloque de estructura residual, donde solo unos pocos dentro del mapeo de identidad, necesitan hacer una coincidencia de cotas y usar una capa convolucional de 1x1 para agregar cotas.

Figura 8.

Arquitectura ResNet50



2. Resultados

En esta sección, se detallan los resultados obtenidos al entrenar y evaluar tres modelos de redes neuronales: YOLOv8, VGG y ResNet50. Estos modelos fueron entrenados utilizando la misma base de datos, la cual consta de un total de 893 imágenes. De estas, 659 imágenes fueron utilizadas para el entrenamiento (train) y 234 imágenes para la validación (val) con un porcentaje del 74% para entrenamiento y un 26 % de validación aproximadamente.

La base de datos se compone de fotografías de dos clases, como se muestra en la Figura 9. Las imágenes fueron capturadas frontalmente, variando la inclinación del rostro de 0° a 90° y de 90° a -90° en el eje X. Además, se considerarán diferentes condiciones de luminosidad, gestos faciales y estilos de peinado, con el fin de la generalización de los modelos.

Figura 9.

Base de datos

Posteriormente, se detalla el entrenamiento de cada uno de los modelos YOLOv8, ResNet50 y VGG, junto con sus respectivas gráficas y pruebas de funcionamiento.

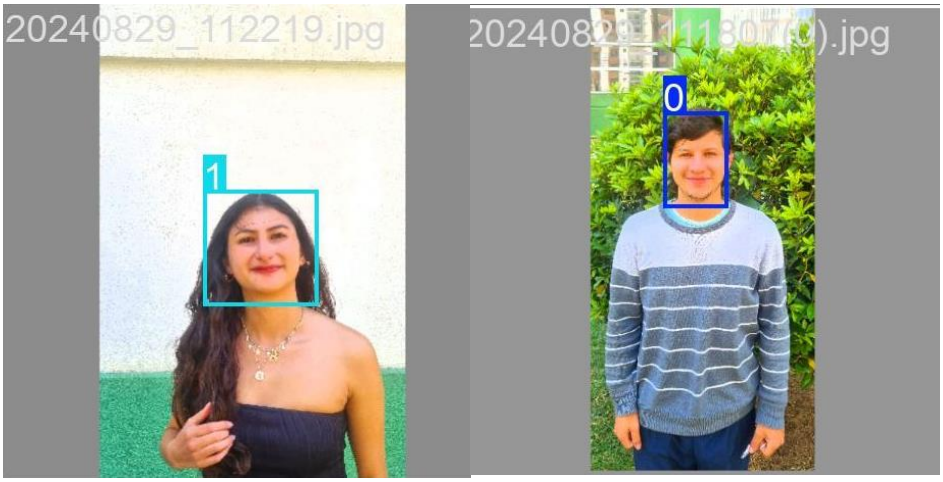
2.1 Resultados Yolov8

El entrenamiento del modelo YOLOv8 se realizó utilizando una base de datos organizada en carpetas “images” y “labels”, En la carpeta de “Imagen” se encuentran alojadas las fotos en formato JPG con una dimensión de 640x640 (requerimiento de Ultralytics), en la carpeta de “labels” se encuentran alojados las etiquetas en formato txt de cada una de las fotos teniendo en cuenta la clase y la dimensión del bounding box. Para realizar dicha tarea se utilizó la herramienta “LabelImg” con la finalidad de darle a cada clase la etiqueta correspondiente.

Terminada la tarea de asignación de etiquetas se pasa a entrenar el modelo, el cual se realizó con 60 épocas, las cuales tuvieron un tiempo de ejecución de 0,223h. Los resultados, mostrados en la Figura 10, presentan ejemplos visuales del rendimiento del modelo YOLOv8. En esta representación, se observan las imágenes procesadas por el algoritmo, donde cada rostro detectado está enmarcado por un rectángulo azul. A cada detección se le asigna una etiqueta numérica: ‘0’ para el rostro de Alejandro y ‘1’ para el de Vanessa.

Figura 10.

Reconocimiento facial mediante el modelo de Yolov8 etiquetas

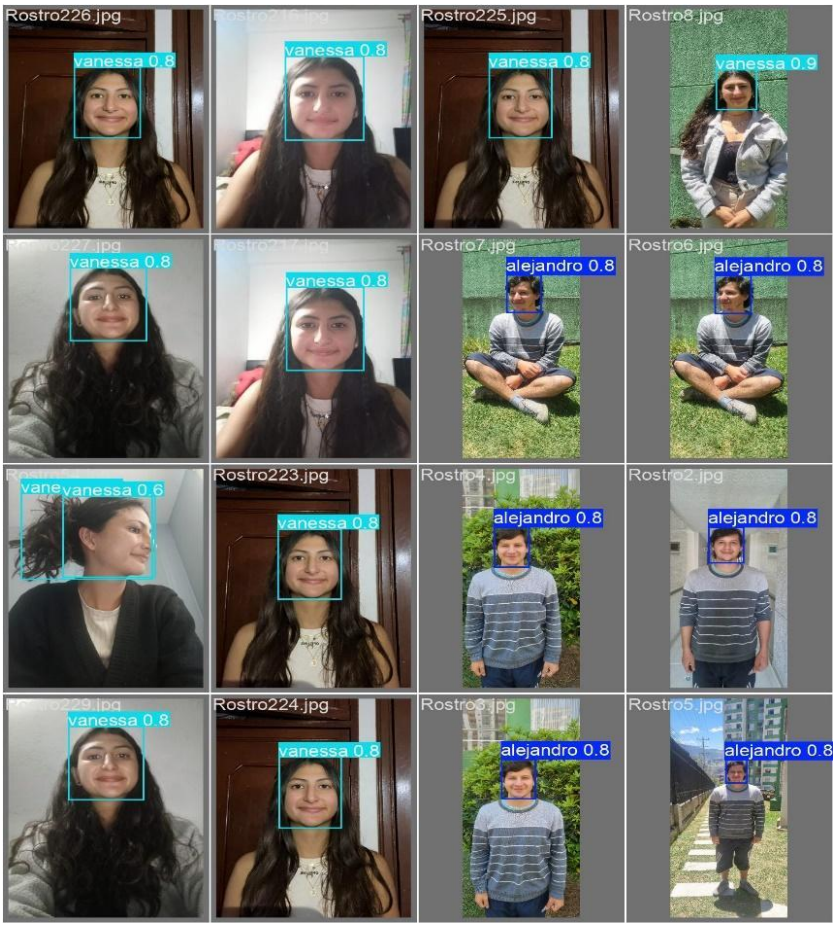


A continuación, en la figura 11 se indican las dos clases “Vanessa” y “Alejandro” y se observa que el modelo pudo identificar correctamente las dos etiquetas aun en condiciones de baja y alta luminosidad, pose del rostro y la distancia focal.

En la figura 11 se observa también el bounding box con el porcentaje de presión correspondiente. (aproximadamente un 80% para cada clase)

Figura 11.

Reconocimiento facial mediante el modelo de Yolov8

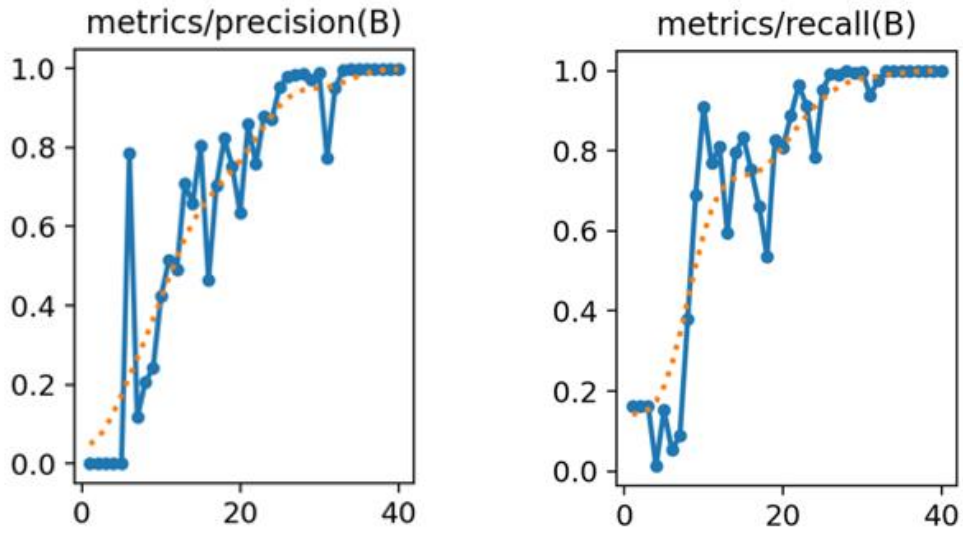


Posteriormente al entrenamiento, se genera una carpeta con los resultados gráficos y numéricos, los cuales se analizarán a continuación. En la Figura 12 se muestran las métricas de precisión y Recall del modelo, En la gráfica de precisión se indica la curva de entrenamiento en función de las épocas. Se observa una tendencia positiva, lo que indica que la precisión del modelo mejora de manera constante a medida que avanzan las épocas. Pasada la época 40, el modelo tiene una convergencia lo que hace que el entrenamiento se detenga para evitar el sobreajuste. Ya que no se obtendrán mejoras sustanciales en su rendimiento.

En la gráfica de Recall se muestra la capacidad del modelo para identificar correctamente las instancias positivas que pertenecen a las clases correspondientes. Esta métrica refleja que las instancias positivas reales son detectadas correctamente por el modelo y aumenta de manera proporcional al paso de las épocas. Aunque se observa que entre las épocas 10 y 20 hay valores que el algoritmo clasifica como negativos. Pero estos se van corrigiendo a medida del tiempo.

Figura 12.

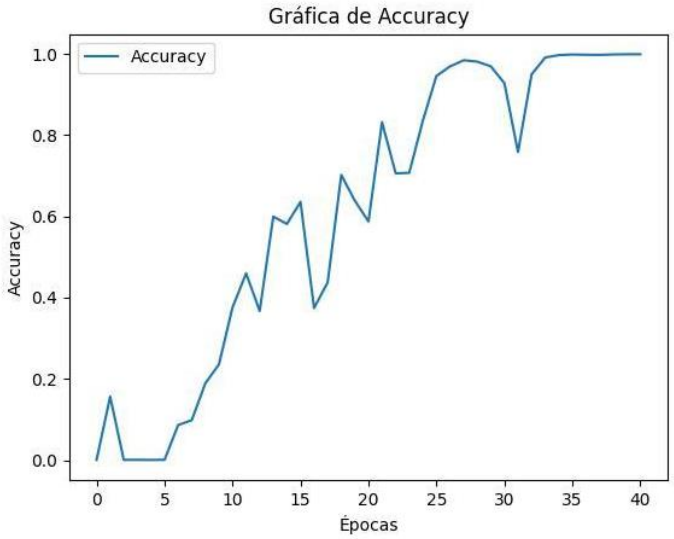
Gráficas de precisión y Recall



En la Figura 13 se observa el entrenamiento en función de la confianza durante las 60 épocas de las cuales se indican solo 40 épocas evitando así el sobreajuste. Se analiza que el comportamiento de los datos fluctúan mucho en las primeras épocas, y se estabiliza dando una confianza superior al 98%.

Figura 13.

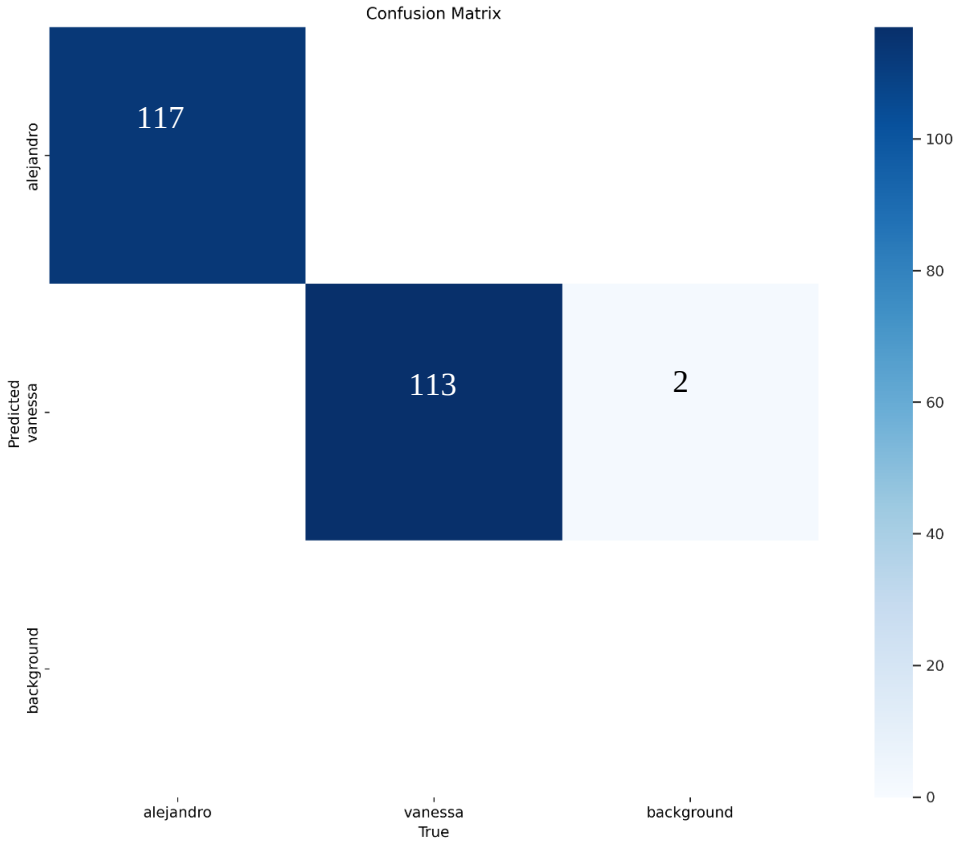
Gráfica Accuracy YoloV8



La matriz de confusión Figura 14 es una herramienta que compara las categorías reales con las predicciones del modelo.

Figura 14.

Gráfica de matriz de confusión

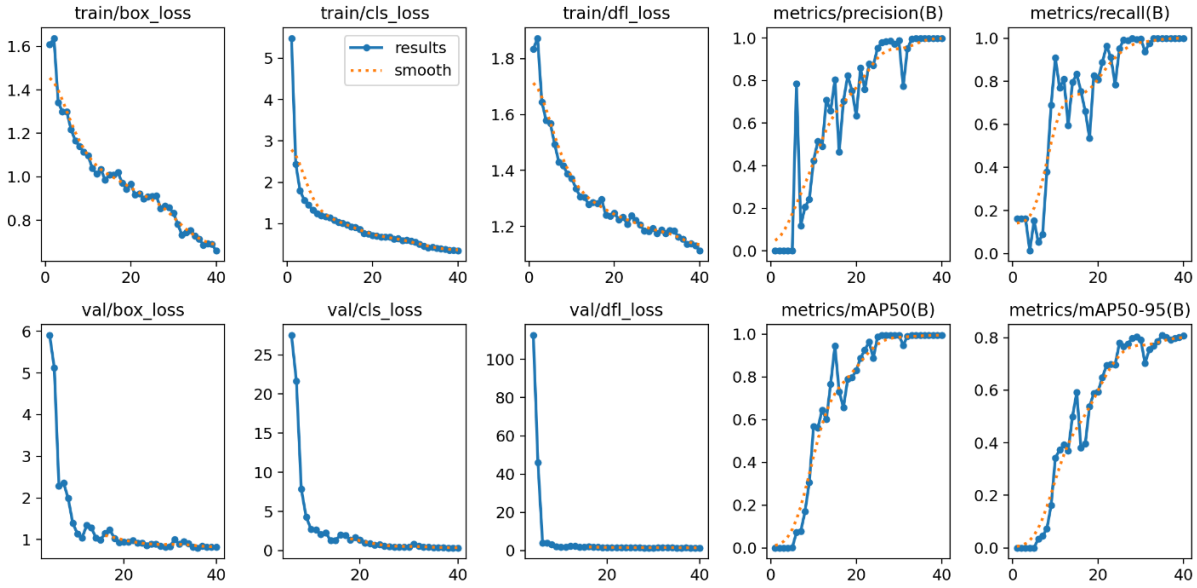


La gráfica indica que el modelo tiene una alta precisión en la clasificación de las categorías “alejandro” y “vanessa.” Y se observa también un mínimo error en donde la clase “vanessa” fue incorrectamente predica como un fondo

En la figura 15 se observa un resumen de todos los resultados arrojados por el algoritmo Yolo v8 en su entrenamiento, estos valores indican una presión superior al 88%, una pérdida por debajo de 1% y un Recall de casi un 98%

Figura 15.

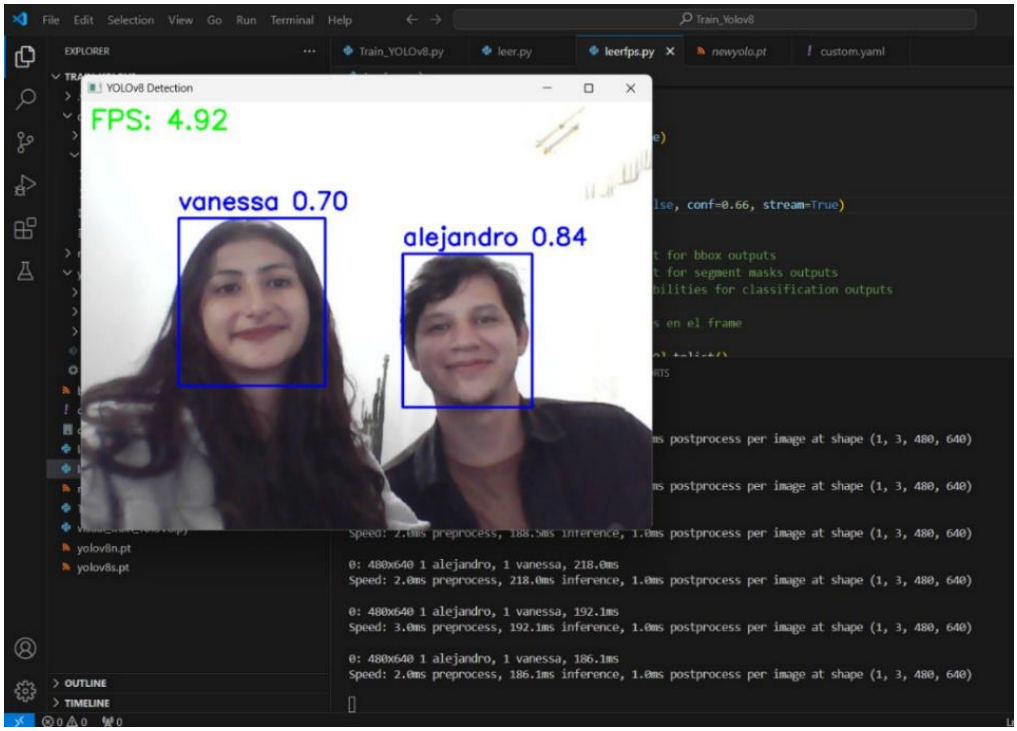
Gráfica de resultados del modelo



En la Figura 16 se observa que el modelo clasifica las dos clases de manera correcta como se asignaron en el entrenamiento. Se observa un cuadro delimitador que contiene la etiqueta, el porcentaje de precisión. Y a un costado se observa la cantidad de FPS de una transmisión de video en tiempo real.

Figura 16.

Clasificación del modelo en tiempo real



2.2 Resultados VGG

Para el entrenamiento del modelo VGG, se utilizó la misma base de datos pero con una modificación de las carpetas. Ya que este modelo no requiere “Labels” o etiquetas con bounding box. Las carpetas se dividen en entrenamiento y validación y dentro de cada una, las clases por separado. Las imágenes se dimensionan a 224x224 con el fin de disminuir el procesamiento de estos datos. El modelo se entrenó con 60 épocas y tuvo un tiempo de ejecución de 3.20h.

En la Figura 17 y Figura 18 se indican las métricas obtenidas en el entrenamiento, las cuales indican la precisión del modelo, la pérdida durante el entrenamiento y la validación de las etiquetas. Estos valores indican que el modelo llegó a una precisión de alrededor del 60%. una pérdida por debajo de 1% y un Recall del 62%.

Figura 17.

Gráfica de resultados del modelo VGG, Presicion y Recall

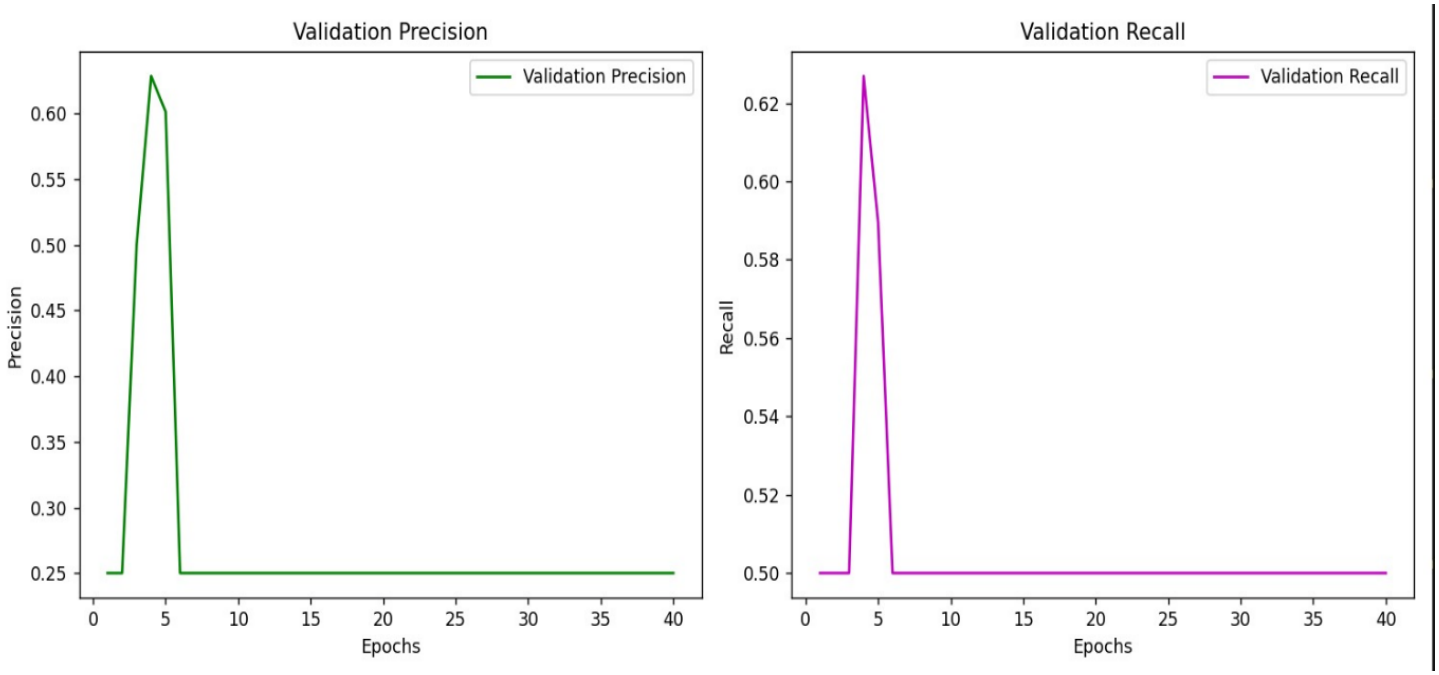
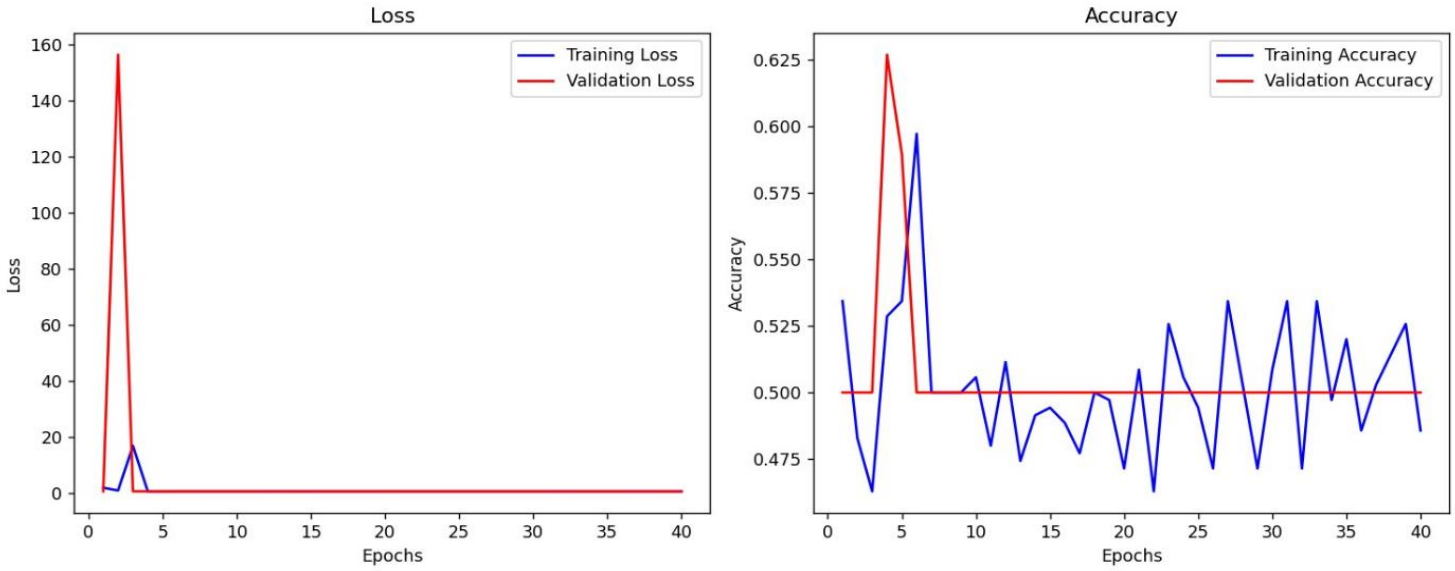


Figura 18.

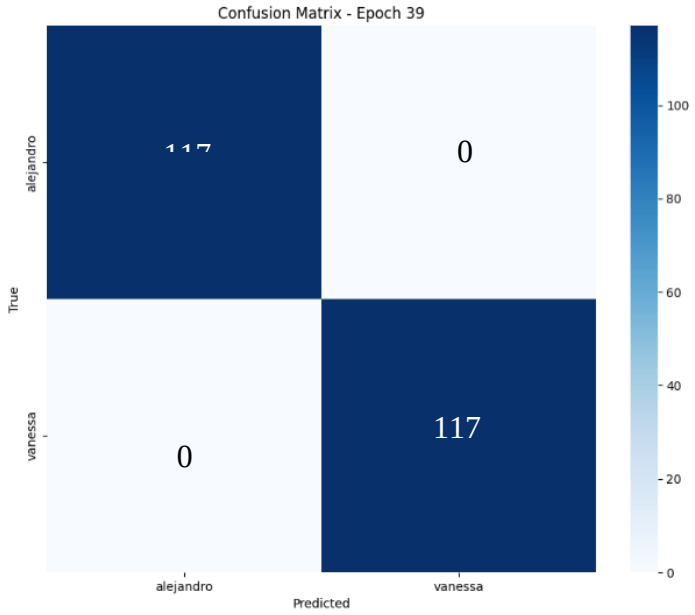
Gráfica de resultados del modelo VGG perdida y confianza



La matriz de confusión compara las categorías reales con las predicciones del modelo. La Figura 19 indica que el modelo tiene una alta precisión en la clasificación de las categorías “alejandro” y “vanessa.” En este caso las categorías son proporcional en ambas clases y no se detectan falsos positivos o fondos.

Figura 19.

Matriz de confusión VGG

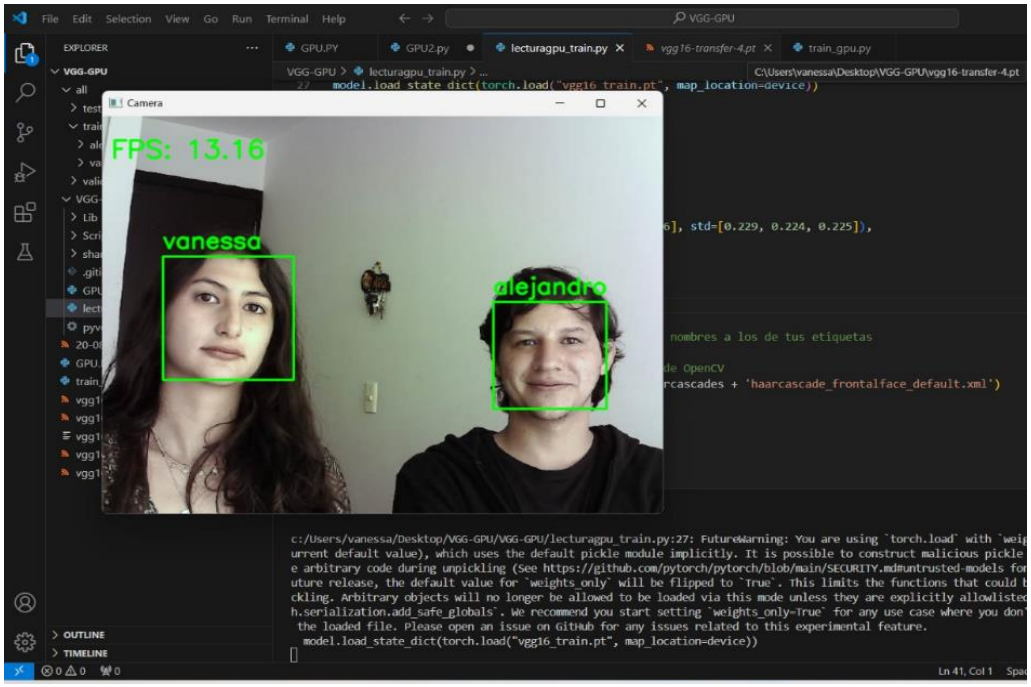


Se guardó los datos del entrenamiento en un archivo de Pytorch generado durante el entrenamiento. Se implementó un código en Python que a través de una cámara web asigna etiquetas a los rostros detectados.

En la Figura 20 se muestra la detección de rostros junto con sus respectivas etiquetas. En un costado del cuadro se indica la tasa de fotogramas por segundo (FPS), la cual fluctúa entre 10 y 15 FPS, un valor considerablemente alto para la detección de rostros.

Figura 20.

Gráfica de resultados del modelo VGG



2.3 Resultados ResNet 50

Para el modelo de ResNet50, se tiene en cuenta la arquitectura del mismo. Se le conoce como bloques residuales, que usa un salto de la variable de entrada, lo que permite al sistema sumar la entrada original a la salida (Team, K. (s. f.-a)). En la serie de 50 capas convoluciones que tiene. La fórmula básica de este es:

Ecuación 13.

Fórmula básica del modelo Resnet-50

$$H(x) = F(x) + x \tag{13}$$

Donde: $H(x)$ Es la salida del bloque residual.

$F(x)$ es la función de las capas de convolución

x es la entrada original.

Este modelo fue entrenado con la misma base de datos. Las carpetas se organizaron por entrenamiento y validación, y dentro de ellas sus respectivas clases. Para el modelo de ResNet 50 se tuvo en cuenta el cálculo de la media y la desviación estándar para el dataset mostrados en la Tabla 9, puesto que es importante para normalizar los datos de entrada. La media se obtuvo calculando la suma de todos los valores del conjunto de datos y viviéndola entre la cantidad de valores. Y la desviación estándar mide cuánto se dispersan los valores con respecto a la media. Esto se hace con las imágenes de la base de datos para dichos cálculos y se indica el cálculo en la Tabla 9.

Ecuación 14.

Media

$$X = \sum \frac{xi * fi}{n} \tag{14}$$

Ecuación 15.

Desviación estándar

$$S = \sqrt{\frac{\sum(x - \underline{x})^2}{n - 1}} \tag{15}$$

A continuación, en la Tabla 9 se muestran los resultados obtenidos al calcular la media y la desviación estándar.

Tabla 9.

Cálculo de media y desviación estándar

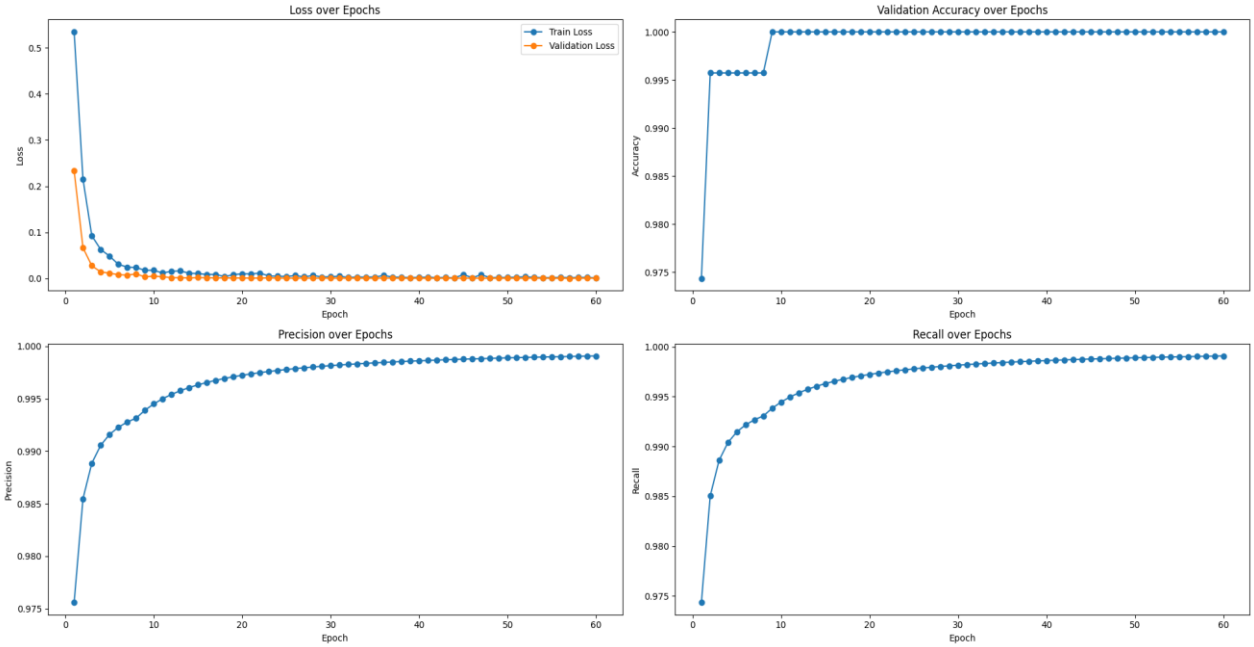
	Tensor
Media	0.4904, 0.4663, 0.3960
Desviación estándar	0.2406, 0.2451, 0.2296

El modelo demuestra un buen entrenamiento en todas las métricas propuestas. La pérdida tanto en el conjunto de entrenamiento como en el de validación disminuye rápidamente y se estabiliza cerca de cero después de aproximadamente 10 épocas, lo que indica que el modelo ha aprendido de manera eficiente. La exactitud de validación alcanza casi el 100% rápidamente.

La precisión y el *Recall* como se observan en la Figura 21 también se estabilizan cerca del 100%, lo que significa que el modelo identifica correctamente casi todas las instancias de las clases sin muchos falsos positivos o falsos negativos.

Figura 21.

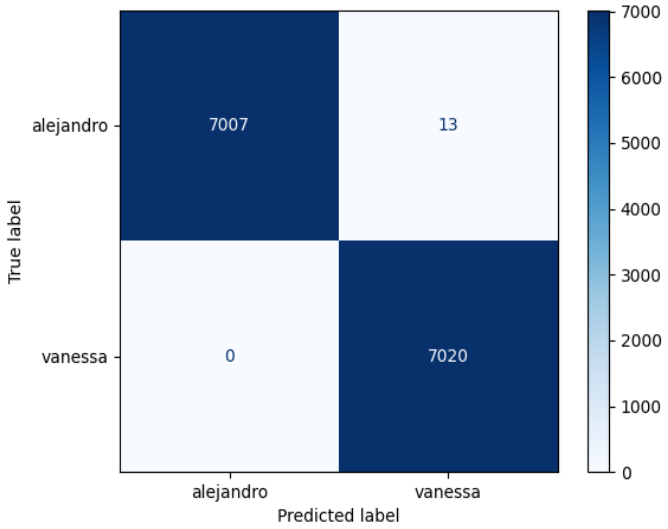
Gráfica de resultados del modelo ResNet50



La matriz de confusión en la Figura 22 compara las predicciones del modelo con los valores reales, con un eje representando la etiqueta verdadera y el otro representando la etiqueta predicha las etiquetas son “alejandro” y “Vanessa”. Se observa en la matriz que para “alejandro”, hay 7007 instancias correctamente predichas como “alejandro” y 13 instancias incorrectamente predichas como “vanessa”. Para “vanessa”, no hay instancias incorrectamente predichas como “alejandro” y 7020 instancias correctamente predichas como “vanessa”.

Figura 22.

Matriz de confusión ResNet50



En la figura 22 se puede confirmar con lo dicho anteriormente que el modelo detectó correctamente las clases con un buen rendimiento, con un porcentaje muy bajo en la predicción de la etiqueta “alejandro” de la verdadera clase con respecto a la predicha.

Después de evaluar los algoritmos, se presenta en la tabla 10 la comparativa de las métricas obtenidas en el entrenamiento de dichos modelos, con el fin de elegir el modelo con las mejores características y métricas para su implementación en la tarjeta Nvidia Jetson Nano, destinada al monitoreo de un hogar.

Tabla 10.

Tabla comparativa de métricas

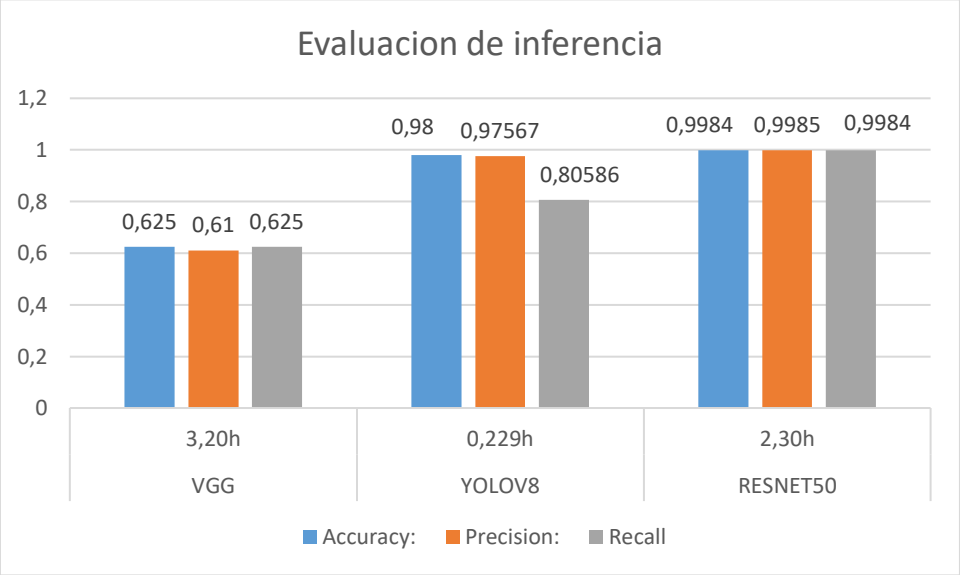
Modelo	Tiempo de ejecución	Épocas	Accuracy:	Precisión:	Recall
VGG	3,20h	60	0,5	0,5	0,5

YOLOV8	0,229h	60	0,98	0,97567	0,80586
RESNET50	2,30h	60	0.9984	0.9985	0.9984

La Figura 23 representa una evaluación de inferencia para los tres modelos donde se compara su *Accuracy, Precisión y Recall.*

Figura 23.

Inferencia de los modelos



Basándonos en los resultados presentados en la Figura 23, YOLOv8 supera a los modelos VGG y RESNET50 en términos de Accuracy, precisión y Recall, mientras también es más eficiente en tiempo de inferencia.

Las Figuras 24, 25 y 26 indican un resultado gráfico de la comparación de los tres algoritmos presentados anteriormente en función de las métricas, precisión, Recall y Accuracy. Donde se puede apreciar de mejor manera el comportamiento de la curva de cada modelo.

Figura 24.

Gráfica de los tres modelos de la métrica "precisión"

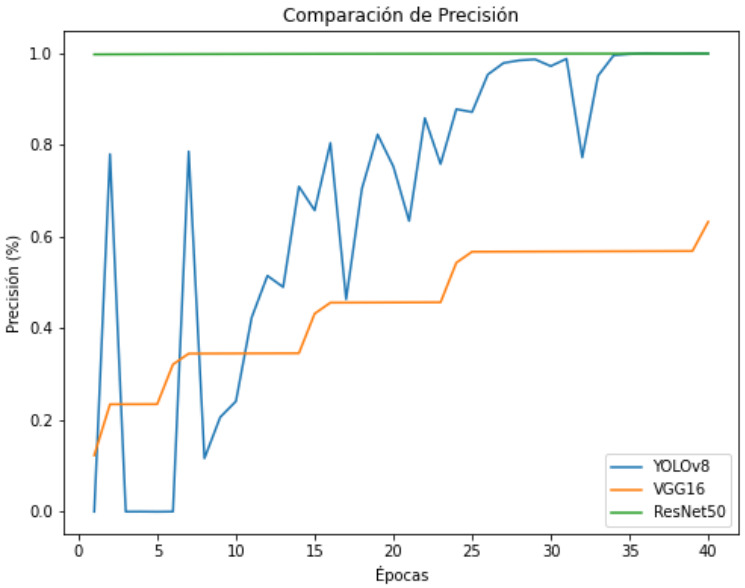


Figura 25.

Gráfica de los tres modelos de la métrica "Recall"

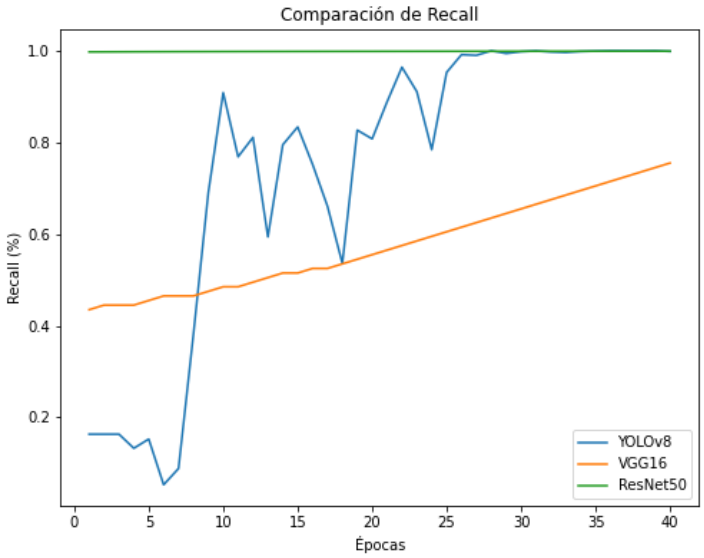
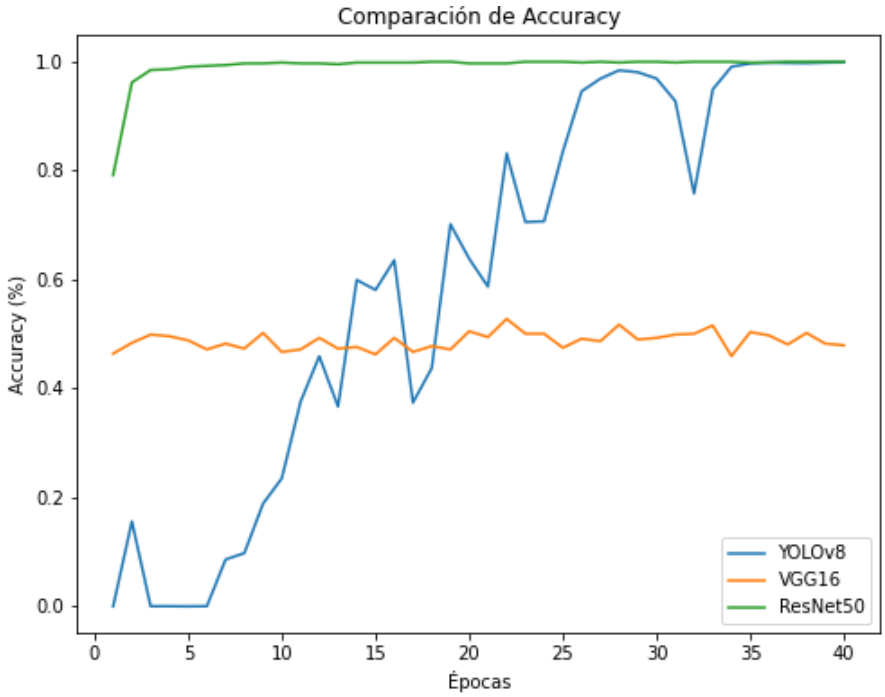


Figura 26.

Gráfica de los tres modelos de la métrica "Accuracy"



Para la implementación en la tarjeta Jetson nano de Nvidia se tuvo en cuenta la configuración básica. El jetpack más actual con soporte de Nvidia es el SDK 4.6.1 el cual cuenta con varias herramientas que se configuran para el procesamiento del modelo. Este Jetpack cuenta con las librerías necesarias como Cuda, Tensor Flow, y cuenta con una versión de Ubuntu 20.0.4. Se instaló en una memoria microSD con una capacidad de lectura de 130 Mb/s y escritura de 100Mb/s. Se configuró un entorno virtual con el fin de mantener librerías necesarias para el modelo de YoloV8, y se realizó la instalación de la herramienta Docker para gestionar los requerimientos necesarios en cada modelo, con el fin de no interferir entre versiones de los mismos.

Se realizó la compra de un ventilador para la disipación del calor generado por el procesador en las diferentes tareas y por lo que el dispositivo genera una alerta

Figura 27.

Configuración del kit de desarrollo Jetson nano

Cómo se puede observar en las Figuras 28, 29 y 30 la ejecución de los modelos en la tarjeta Jetson nano permitió la detección de las dos clases de manera correcta.

Figura 28.

Modelo YoloV8 aplicado en Jetson nano

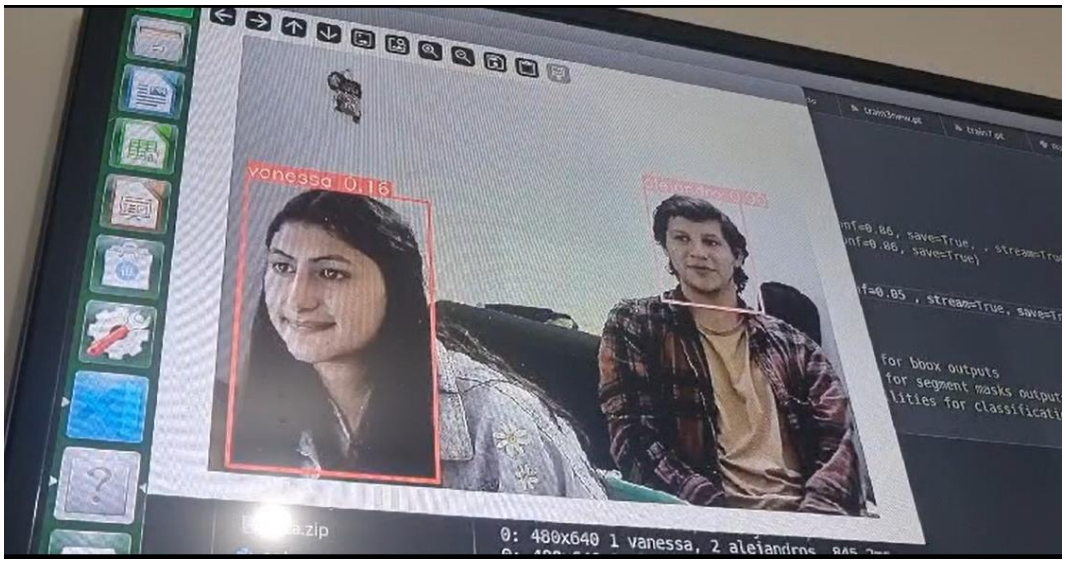


Figura 29.

Modelo VGG aplicado en Jetson nano

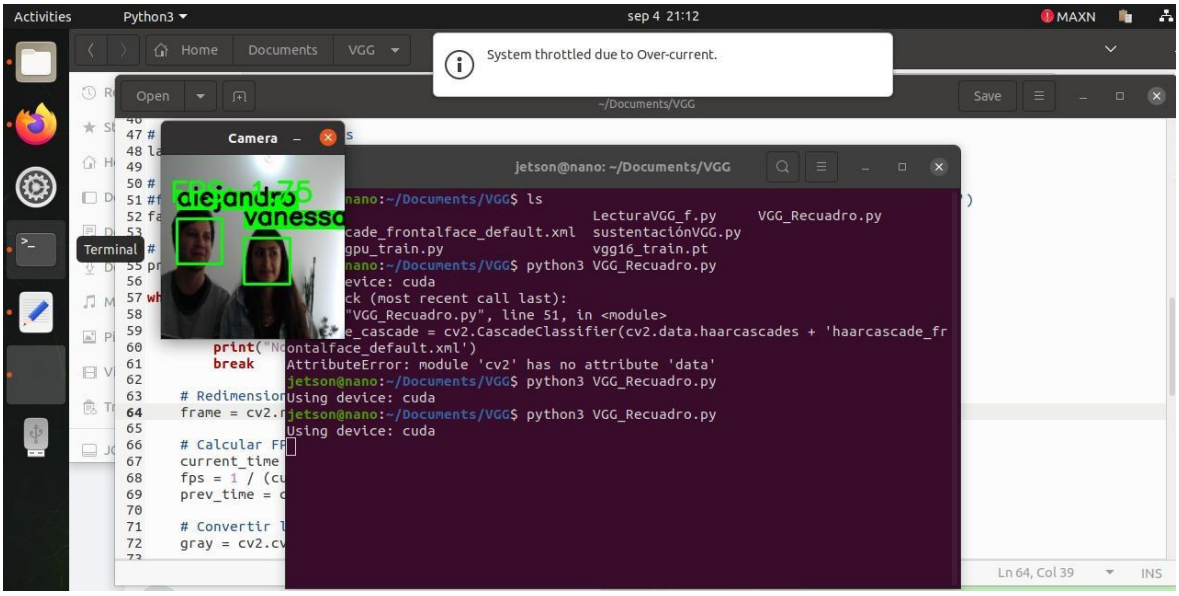
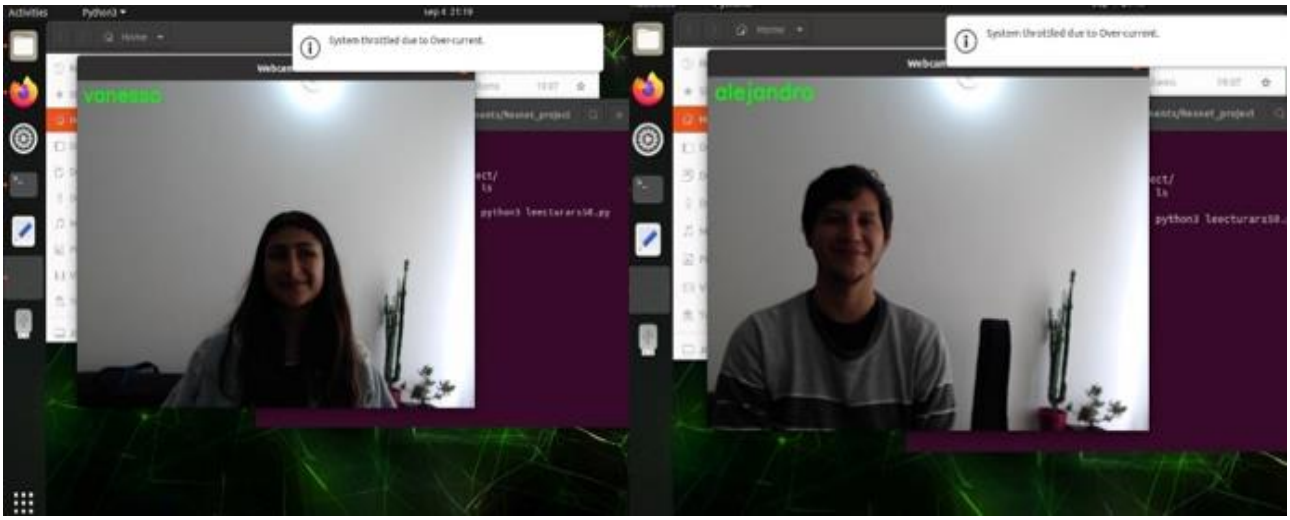


Figura 30.

Modelo Resnet50 aplicado en Jetson nano



3. Conclusiones

Este proyecto ha demostrado que, mediante la selección cuidadosa de algoritmos, el entrenamiento especializado y la optimización en plataformas de Edge Computing, es posible desarrollar un sistema de reconocimiento facial altamente eficiente y seguro. Al evaluar los algoritmos VGG, YOLOv8 y ResNet50 en un kit de desarrollo de Jetson Nano, se comprobó que cada modelo ofrece ventajas específicas en términos de precisión, Recall, matriz de confusión, Accuracy y tiempo de ejecución. El impacto potencial del proyecto a corto plazo, los resultados de este proyecto podrían acelerar la adopción de tecnologías de Edge Computing en aplicaciones de seguridad y reconocimiento facial, ofreciendo soluciones más rápidas y seguras que aquellas basadas en procesamiento en la nube. A largo plazo, el impacto potencial incluye la integración de sistemas de reconocimiento facial más confiables y eficientes en una variedad de contextos, desde la seguridad en el hogar hasta la vigilancia en entornos comerciales.

Factores críticos como la iluminación, la pose y las variaciones en los datos de entrada (como peinados y expresiones faciales) fueron identificados como elementos que pueden influir en el rendimiento de los algoritmos. La incorporación de un análisis detallado de estos factores y su mitigación mediante técnicas de preprocesamiento y ajustes en los modelos ha demostrado ser esencial para mejorar el sistema.

En la tarjeta Jetson nano el uso de Docker para la gestión de entornos permitió evitar conflictos de versiones y facilitó la implementación de los modelos en la plataforma, lo cual es necesario para garantizar la estabilidad del sistema en un entorno real. Este proyecto genera las bases para futuras investigaciones y mejoras en el campo de la seguridad digital y el monitoreo en entornos domésticos y empresariales. El impacto social, ético y económico de este proyecto también es significativo. Desde un punto de vista social, una mayor precisión en el reconocimiento facial puede mejorar la seguridad personal y comunitaria, reduciendo el riesgo de falsos positivos y errores de identificación. Sin embargo, también es necesario considerar los aspectos éticos, como la privacidad y el sesgo algorítmico, que deben ser abordados para evitar desigualdades y proteger los derechos individuales.

Referencias

Antona-Cortés, C. (2007). *Herramientas modernas en redes neuronales: la librería Keras.* Editorial UAM. Departamento de Ingeniería Informática

Gortázar-Bellas, F., Martínez-Unanue, R., y Fresno-Fernández, V. (2016). *Lenguajes de programación y procesadores.* Scribd. https://es.scribd.com/document/470290483/Lenguajes-de-Programacion-y-Procesadores-Francisco-Gortazar-Bellas-Raquel-Martinez-Unanue-Victor-Fresno-Fernandez

Graveto, V., Cruz, T., y Simöes, P. (2022). Security of Building Automation and Control Systems: Survey and future research directions. *Rev. Computers & Security, 112.*

Github. (2023). *Ultralytics. YOLOv8: Real-Time Object Detection.* Github. https://github.com/ultralytics/ultralytics

Han, H., Shan, S., Chen, X., y Gao, W. (2013). A comparative study on illumination preprocessing in face recognition. *Rev. Pattern Recognition, 46(6), 1691-1699.*

Igual, R., y Medrano, C. (2008). *Tutorial de OpenCV.* Academia.edu. https://www.academia.edu/29801907/Tutorial_de_OpenCV

Keras. (s.f.). *ResNet and ResNetV2.* Keras documentation. https://keras.io/api/applications/resnet/

Kowalski, M. Ł., y Grudzień, A. (2018). High-resolution thermal face dataset for face and expression recognition. *Rev. Metrology and Measurement Systems, 25, 403–415.* https://doi.org/10.24425/119566

Lindner, T., Wyrwał, D., Białek, M., y Nowak, P. (2020). *Face recognition system based on a single-board computer* (Presentacion de la conferencia). International Conference Mechatronic Systems and Materials (MSM).

Lluch-Crespo, J. (2022). *Introduccion a la librería Pandas.* Universitat Politecnica de Valencia. https://riunet.upv.es/handle/10251/183074?show=full

Lundberg, S. (2018). *Welcome to the SHAP documentation.* Shap. https://shap.readthedocs.io/en/latest/

Molina-Garrido, D. A. (2019). *Clasificación de género con análisis racial en imágenes de espectro visual mediante técnicas de Deep Learning.* S.n.

Perez-Prieto, J. A. (2019). *La librería científica Scipy.* Reserach,iac.es. https://research.iac.es/sieinvens/python-course/scipy.html

Pertuz, C. M. P. (2022). *Aprendizaje automático y profundo en python*. Editorial Ra-Ma.

Policia Naciional de Colombia-PONAL. (2022). *Estadistica delictiva.* PONAL.

Rojas, E. M. (2020). Machine Learning: análisis de lenguajes de programación y herramientas para desarrollo. *Rev. Ibérica de Sistemas e Tecnologias de Informação, (E28), 586-599.*

Rouhiainen, L. (2018). *Inteligencia artificial 101 cosas que debes saber hoy sobre nuestro futuro.* Editorial Alienta.

Sánchez-Alberca, A. (2016a). La librería Matplotlib. *Blog Aprende con Alf.* https://aprendeconalf.es/docencia/python/manual/capitulo-matplotlib/

Sánchez-Alberca, A. (2016b). La librería Numpy. *Blog Aprende con Alf.* https://aprendeconalf.es/docencia/python/manual/capitulo-numpy/

Santana, M. A. G., Díaz-Sánchez, L. E., Paz, I. T., y Huertas, M. R. (2017). Estado del arte en reconocimiento facial. *Rev. Res. Comput. Sci., 140, 19-27.*

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Rev. Neural Networks, 61, 85-117.* doi: 10.1016/j.neunet.2014.09.003

Shetty, A., Bhoomika, E., Deeksha, B., Rebeiro, J., y Ramyashree, B. (2021). Facial recognition using Haar cascade and LBP classifiers. *Rev. Global Transitions Proceedings, 2(2), 330-335.*

Socolinsky, D. A., y Selinger, A. (2002). A comparative analysis of face recognition performance with visible and thermal infrared imagery. *Rev. International Conference on Pattern Recognition, 4, 217-222.*

Tahir, M. F., y Saqib, M. A. (2016). Optimal scheduling of electrical power in energy-deficient scenarios using artificial neural network and Bootstrap aggregating. *Rev. International Journal of Electrical Power & Energy Systems, 83, 49-57.*

UniPython. (2021). Detección de rostros, caras y ojos con haar cascad. *Blog UniPython.* https://unipython.com/deteccion-rostros-caras-ojos-haar-cascad/

Universidad De Alcalá. (2022). *Scikit-learn, herramienta básica para el data science en python.* Universidad de Alcala. https://www.master-data-scientist.com/scikit-learn-data-science/

Rozada Raneros, S. (2021). Estudio de la arquitectura YOLO para la detección de objetos mediante deep learning.

Villalba, M. (2020, October 26). Arquitectura VGG16 y VGG19 en Deep Learning. https://keepcoding.io/blog/arquitectura-vgg16-vgg19-deep-learning/

K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90. keywords: {Training;Degradation;Complexity theory;Image recognition;Neural networks;Visualization;Image segmentation},

Valle-Barrio, A. (2018). *Aplicación de Tensorflow en deep learning* (Tesis de magister, Universidad Politecnica de Madrid).

Zou, J., Ji, Q., y Nagy, G. (2007). Un estudio comparativo del enfoque de coincidencia local para el reconocimiento facial. *Rev. IEEE Transactions on image processing, 16(10), 2617-2628.*

Anexos

Anexo A. Cronograma

		Meses													
Objetivo	**Actividad**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Identificar y analizar los criterios de autentificación biométrica y aprendizaje automático basado en reconocimiento de rostros.	Investigar y evaluar diferentes sistemas de autenticación biométrica basados en reconocimiento facial disponibles en el mercado. Compara sus características, precisión, velocidad.														
	Realizar pruebas de campo de los sistemas seleccionados para evaluar su desempeño en situaciones reales. Recopila datos sobre la precisión de la autenticación, la velocidad de respuesta y la robustez														

	Analizar los resultados de las pruebas y seleccionar el sistema de autenticación biométrica basado en reconocimiento facial que mejor se ajuste a los criterios de precisión, velocidad y seguridad, así como los costos asociados con la implementación y el mantenimiento del sistema.	
Evaluar y comparar diferentes algoritmos de identificación, extracción y clasificación de características para la detección de rostros en video basado en métodos de aprendizaje automático supervisado	Implementar y entrenar diferentes algoritmos de detección de rostros en video.	
	Evaluar el rendimiento de los algoritmos en diferentes condiciones de iluminación y oclusión facial.	
	Realizar pruebas de rendimiento en tiempo real y comparar la velocidad de detección.	
Realizar una evaluación estadística		

de los algoritmos en una plataforma de Edge Computing a través de un conjunto de datos para la identificación de rostros, considerando factores como la precisión, robustez y repetibilidad del sistema	Configuración de la plataforma de Edge Computing	
	Implementación y evaluación de los algoritmos seleccionados en la plataforma de Edge Computing con los resultados obtenidos se realiza una evaluación estadística comparativa basada en los factores de precisión, robustez y repetibilidad.	
	Analiza los resultados obtenidos y realiza comparaciones entre los algoritmos evaluados.	

Anexo B. Recursos y presupuestos

RUBROS	FUENTES			TOTAL
	ESTUDIANTES	PROGRAMA	JUSTIFICACIÓN	
Personal				
Equipo	Computador		Este elemento es necesario para entrenar modelos, visualizarlos y graficar las métricas, así como gestionar la documentación del proyecto de grado	$ 3’000.000,00
Software	● (Visual Studio code licencia MIT) ● (Python Software Foundation License)		El entorno de visual studio code es esencial para gestionar las bibliotecas y poder utilizar la extensión de Python y aplicación de los algoritmos	0
Tarjeta SD Samsung pro plus			Estas tarjetas SD son necesarias para cargar la imagen	150.000

Tarjeta SD CANVAS gol plus			ISO para la configuración y el funcionamiento de la jetson nano	70.000
Ventilador Fan jetson nano			Este dispositivo es necesario para evitar el sobrecalentamiento que se genera al ejecutar los modelos	200.000
Materiales		● Tarjeta Nvidia Jetson nano ● Cámaras		$3’700.000
Servicios técnicos				
Publicaciones	Participación en el cuarto congreso de microcontroladores y sus aplicaciones			
Administración				
Comunicaciones				

Otros	
TOTAL	7.120.000