UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES

Tema:

SISTEMA DE DETECCIÓN DE SITUACIONES DELICTIVAS EN ESTABLECIMIENTOS COMERCIALES USANDO INTELIGENCIA ARTIFICIAL

Trabajo de titulación modalidad Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en telecomunicaciones

ÁREA: Comunicaciones

LÍNEA DE INVESTIGACIÓN: Tecnología de la información y Sistemas de control.

AUTOR: Erick Fabian Sandoval Robayo

TUTOR: Ing. Ana Pamela Castro Martin MSc.

Ambato - Ecuador

febrero 2024

APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA DE DETECCIÓN DE SITUACIONES DELICTIVAS EN ESTABLECIMIENTOS COMERCIALES USANDO INTELIGENCIA ARTIFICIAL, desarrollado bajo la modalidad Proyecto de Investigación por el señor Erick Fabian Sandoval Robayo, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.3 del instructivo del reglamento referido.

Ambato, febrero 2024.

Firmado electrónicamente por:
ANA PAMELA CASTRO MARTIN

Ing. Ana Pamela Castro Martin, MSc.

TUTOR

AUTORÍA

El presente trabajo de titulación con el tema: SISTEMA DE DETECCIÓN DE SITUACIONES DELICTIVAS EN ESTABLECIMIENTOS COMERCIALES USANDO INTELIGENCIA ARTIFICIAL es absolutamente original, auténtico y personal y ha observado los preceptos establecidos en la Disposición General Quinta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, febrero 2024.

Erick Fabian Sandoval Robayo

C.C. 1803830668

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que reproduzca total o parcialmente este trabajo de titulación dentro de las regulaciones legales e institucionales correspondientes. Además, cedo todos mis derechos de autor a favor de la institución con el propósito de su difusión pública, por lo tanto, autorizo su publicación en el repositorio virtual institucional como un documento disponible para la lectura y uso con fines académicos e investigativos de acuerdo con la Disposición General Cuarta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato.

Ambato, febrero 2024.

Erick Fabian Sandoval Robayo

C.C. 1803830668

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por el señor Erick Fabian Sandoval Robayo, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA DE DETECCIÓN DE SITUACIONES DELICTIVAS EN ESTABLECIMIENTOS COMERCIALES USANDO INTELIGENCIA ARTIFICIAL, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con la señora Presidente del Tribunal.

Ambato, febrero 2024.

Firmado electrónicamente por:
ELSA PILAR URRUTIA URRUTIA

Ing. Elsa Pilar Urrutia Urrutia, Mg.
PRESIDENTE DEL TRIBUNAL

Firmado electrónicamente por:
EDGAR PATRICIO CORDOVA CORDOVA

Ing. Edgar Córdova Córdova, Mg.
PROFESOR CALIFICADOR

Firmado electrónicamente por:
GEOVANNI DANILO BRITO MONCAYO

Ing. Geovanni Brito Moncayo, Mg.
PROFESOR CALIFICADOR

DEDICATORIA

El presente proyecto va dedicado a mis amados padres, Álvaro, Anita, y a mis queridos hermanos e hijos de 4 patas, por su amor incondicional y constante apoyo han sido mi mayor fortaleza a lo largo de este fascinante viaje académico. Cada logro alcanzado lleva impreso su sacrificio y dedicación.

A mis leales amigos, por su apoyo inquebrantable y solidaridad han iluminado los momentos más desafiantes. Gracias por estar a mi lado con sus consejos, paciencia y amistad sincera. Su presencia ha sido una fuente constante de inspiración y alegría. Agradezco cada gesto de aliento y celebración compartida, haciendo de este camino una experiencia memorable.

Este logro no solo es mío, sino de todos ustedes que han contribuido a mi éxito de diferentes maneras. Con gratitud eterna, les dedico este trabajo que representa el fruto de nuestro esfuerzo colectivo.

Con cariño y agradecimiento, Erick.

AGRADECIMIENTO

Quisiera expresar mi sincero agradecimiento a todas las personas que han sido esenciales en mi trayecto académico y han contribuido significativamente al éxito de esta tesis.

A mis amados padres, y a mis queridos hermanos, su amor, apoyo incondicional y paciencia han sido mi mayor fortaleza a lo largo de esta intensa travesía. Cada logro alcanzado es un reflejo de su dedicación y sacrificio. Gracias por ser mi fuente constante de inspiración.

Agradezco de manera especial a mi Tutora de tesis, Ing. Pamela Castro por sus valiosas enseñanzas y orientación han iluminado mi camino académico. Cada lección compartida ha sido un regalo que trasciende las páginas de este trabajo. Su influencia ha sido transformadora, y les estoy eternamente agradecida.

A toda mi familia extendida y seres queridos, gracias por celebrar cada pequeño logro, por alentar mis sueños y por ser una fuente constante de amor y apoyo. Su presencia ha sido el fundamento de mi éxito.

Este trabajo no solo es el resultado de mi esfuerzo individual, sino también de la contribución invaluable de cada uno de ustedes. Gracias por formar parte de este capítulo importante de mi vida.

Con gratitud y aprecio, Erick

ÍNDICE GENERAL DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN EJECUTIVO

El incremento constante de la inseguridad en Ecuador es un fenómeno que se manifiesta anualmente mediante una variedad de delitos, siendo los robos con armas en establecimientos comerciales el enfoque principal de la presente investigación. En este contexto, se llevó a cabo un estudio con el propósito de desarrollar un sistema basado en inteligencia artificial para la detección de situaciones delictivas en locales comerciales. El objetivo primordial de este sistema es generar alertas ante posibles asaltos, con la finalidad de agilizar la respuesta por parte de las autoridades competentes.

Este sistema fue hecho con las etapas de adquisición de datos, procesamiento, y visualización. En la fase de adquisición, se emplea una cámara Rohxizw 1520p Hd Wifi para capturar imágenes en tiempo real. Los datos capturados se transmiten al microordenador NVIDIA JETSON NANO para su procesamiento. El sistema fue desarrollado en Python y se optó por Yolov8 como algoritmo de visión artificial, el cual es el encargado de procesar los datos, dando como resultado el reconocimiento de los indicadores asociados al robo con armas. La visualización de los resultados obtenidos y las alertas generadas llegan por Telegram, proporcionando imágenes del delito, una descripción detallada de la situación y la ubicación.

Es relevante destacar que la información es procesada y transmitida en tiempo real, asegurando una respuesta eficiente ante posibles incidentes delictivos. Teniendo como resultado una efectividad del 84% luego de realizar las pruebas correspondientes.

Palabras clave: Robos, inteligencia artificial, Yolov8, armas.

ABSTRACT

The constant increase in insecurity in Ecuador is a phenomenon that manifests annually through a variety of crimes, with armed robberies in commercial establishments being the main focus of the present research. In this context, a study was carried out with the purpose of developing an artificial intelligence-based system for the detection of criminal situations in commercial premises. The primary objective of this system is to generate alerts in the face of potential assaults, with the aim of expediting the response from the relevant authorities.

This system was built in three stages: data acquisition, processing, and visualization. In the acquisition phase, a DS-2CD2147G2 IP camera with 5 megapixels is used to capture real-time images. The captured data is transmitted to the NVIDIA JETSON NANO microcomputer for processing. The system was developed in Python, and Yolov8 was chosen as the artificial vision algorithm, responsible for processing the data and recognizing indicators associated with armed robbery. The visualization of the results and generated alerts is sent through Telegram, providing images of the crime, a detailed description of the situation, and the location.

It is relevant to highlight that the information is processed and transmitted in real-time, ensuring an efficient response to potential criminal incidents. The system achieved an effectiveness of 84% after conducting the corresponding tests.

Keywords: Robberies, artificial intelligence, Yolov8, weapons

CAPÍTULO I. MARCO TEÓRICO

1.1 Tema de investigación

SISTEMA DE DETECCIÓN DE SITUACIONES DELICTIVAS EN ESTABLECIMIENTOS COMERCIALES USANDO INTELIGENCIA ARTIFICIAL

1.1.1 Planteamiento del problema

La tecnología desempeña un papel cada vez más relevante en la vida cotidiana, y es común encontrar sistemas tecnológicos en todo tipo de ámbitos como la medicina, la industria, la seguridad, etc.

Existe una gran variedad de empresas que se especializan en brindar servicios de alarmas y monitoreo. Entre ellas se encuentran líderes del sector de las tecnologías de seguridad, reconocida por su experiencia en seguridad y monitoreo. Estas empresas proporcionan una amplia gama de soluciones de seguridad, que van desde sistemas de alarma básicos hasta sistemas avanzados del mismo. [1]

En Ecuador, los sistemas de seguridad, como las cámaras de seguridad, son ampliamente utilizados debido a su eficacia para combatir la delincuencia. Entre la variedad de opciones disponibles, las cámaras con tecnología de visión artificial se han vuelto especialmente populares. Estas características combinadas brindan una solución completa y confiable para la seguridad al no ser tan solo un sistema que sirva para almacenar datos sino para alarmar en el caso de detectar una anomalía. [2].

En Ambato, la seguridad se ha convertido en una preocupación importante, lo que ha llevado a la creciente demanda de sistemas de alarmas y monitoreo. Tanto los propietarios de negocios como los residentes buscan soluciones efectivas para protegerse contra robos y extorsiones. Procesar mucho tiempo de videovigilancia para una persona es un trabajo muy pesado, por lo cual los sistemas automáticos con IA se han vuelto una herramienta eficaz para esta problemática. Estos sistemas deben contar

con mayor inteligencia para emitir alertas automáticas en casos de personas sospechosas, movimientos inusuales, colisiones u otros eventos. [3]

Los robos, asaltos y extorsiones han aumentado debido a la falta de fuentes de trabajo en Ecuador. Además de tener otras razones como la falta de una respuesta efectiva por parte de las autoridades ha llevado al aumento de bandas criminales y grupos delictivos. La falta de soluciones por parte de las autoridades ha provocado un aumento en las extorsiones, no sólo en locales comerciales, sino también en las escuelas y universidades, generando inseguridad y temor en los ciudadanos. Como resultado, la población busca herramientas para prevenir incidentes instalando sistemas de seguridad.

1.2 Antecedentes investigativos

Se tomaron estudios realizados previamente con relación a la presente investigación, los cuales proporcionarón información relevante para la fundamentación y desarrollo del prototipo.

En 2022 Tamara Radivilova de Kharkiv National University of Radio Electronics , en la investigación “Detection of Shoplifting on Video Using a Hybrid Network” en el cual se ocupó una red convolucional para extraer características de los cuadros por segundo. La red convolucional fue utilizada para procesar cada cuadro individual de los videos examinados. Este proceso de extracción de características implica que la red analiza y comprende las características visuales clave presentes en cada cuadro, capturando detalles que son relevantes para la identificación de conductas asociadas al hurto en tiendas. clasificó los fragmentos de video. Los resultados de este modelo dieron una precisión del 93%. [4]

En 2022 Montesdeoca García, Carlos André de la Escuela Politécnica Nacional, en la investigación “Desarrollo de aplicación móvil de geolocalización para alarma y notificación de pánico en plataforma Android.”, en donde diseñó un aplicativo móvil que permite usar los teléfonos como medio de alerta, el cual ocupó la metodología Scrum que ayuda al crecimiento progresivo del desarrollo y se usó Ionic Framework para agilizar la codificación y para la gestión de datos se ocupó NoSQL. Como

resultado se logró tener una opción rápida y eficaz de alertar con datos de la localización de la persona en peligro. [5]

En 2021 Martinez Guillermo, de la School of Engineering and Sciences, Tecnologico de Monterrey, en la investigación “Criminal Intention Detection at Early Stages of Shoplifting Cases by Using 3D Convolutional Neural Networks.”, en donde se implementó un modelo 3dCNN como extractor de funciones de video, lo que implica aprovechar las capacidades tridimensionales de la red para comprender y analizar patrones temporales en datos visuales en movimiento, lo que resulta valioso en diversas aplicaciones, desde la vigilancia hasta el reconocimiento de acciones y eventos en videos. Se obtuvo como resultado que el modelo clasifica correctamente el comportamiento del sospechoso en la mayoría de los escenarios donde se probó, la cual tiene un valor de precisión de 0.8571. [6]

En 2020 Chicaiza Guachi, Karla Gabriela y Brito Moncayo, Geovanni Danilo en la investigación “Sistema de alarma comunitaria para el mercado San Juan de la Ciudad de Santiago de Píllaro” la cual se implementó en un centro que no cuenta con un sistema adecuado de seguridad y que por medio de la tecnología hace que la comunidad esté alerta de actividades crimínales. En esta investigación se determinó el uso de aplicaciones de mensajería inmediata como WhatsApp y Telegram para tener una respuesta inmediata. Se ocupó una Rasberry Pi 3B+ y una Camara Ip para el monitoreo y activación de las alarmas, se desarrolló todo en el lenguaje programación Python, para la comunicación se utilizó telefonía IP donde se ocupó Asterisk como centralita de extensiones. Se logró tener un sistema funcional con monitoreo y detección de movimiento en donde se pudo mejorar la seguridad del sector en donde se implementó el sistema. [7]

En 2018 Aguilar Villalba, Freddy Rene de la Escuela Superior Politécnica de Chimborazo, en la investigación “Desarrollo de un prototipo de alarma multimodal comunitaria utilizando el protocolo IPv6 y GPRS para Smart Cities con monitoreo en tiempo real.”, en donde se ocupó el protocolo IPv6 y GPRS, se optó por Arduino Mega como microprocesador, para el monitoreo se empleó una Rasberry Pi4, y los sensores utilizados fueron sensores de tipo magnético y ultrasónico analógico. Como resultado

del proyecto se obtuvo un modelo rápido y eficaz para alertar a la comunidad en donde fue implementado. [8]

1.3 Fundamentación teórica

Los aspectos más importantes dentro de la presente investigación son: Procesamiento de imágenes, adquisición de imágenes, visión artificial, tecnologías de seguridad, los cuales son elementos indispensables dentro de la investigación.

1.3.1 Inteligencia artificial

Su objetivo es que las máquinas logren desarrollar las actividades que hacen los seres humanos. Este se clasifica dependiendo del tipo de entrenamiento el cual tiene los siguientes:

a. Machine learning

Área de las ciencias computacionales más aplicada en donde se desarrollan algoritmos donde se recopilan datos para que se pueda extraer características de interés para generar datos de salida para interpretar la información que se ingresa al sistema. Este sistema segmenta, extrae características, que permite diferenciar una característica de otra para que el sistema clasifique, lo que permite determinar a que corresponde y cómo está relacionada con las demás características. En la Figura 1 indica los procesos que tiene los cuales son entrada, segmentación, extracción de datos, clasificación y salida [9].

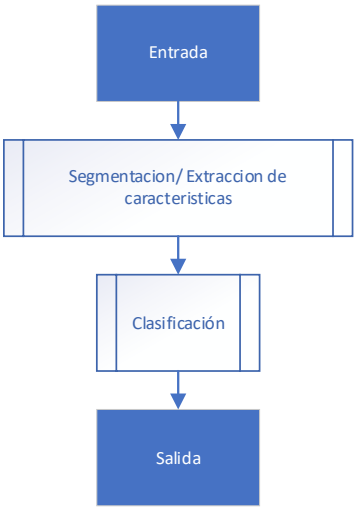


Figura 1. Machine Learning [9]

b. Deep Learning

Este modelo no tiene una etapa de segmentación y extracción de características, este de forma autónoma ya clasifica los datos, mediante un proceso de entrenamiento. Por lo cual ya no se requiere la intervención humana haciéndolo un sistema mucho más autónomo. Según Figura 2 la estructura de Deep Learning es la entrada, clasificación y salida. [9]

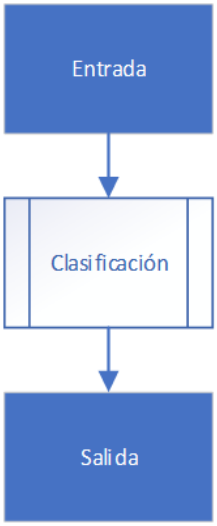


Figura 2. Deep Learning [9]

c. Neurona

Algoritmo computacional, en la Figura 3 se puede ver que toma datos de entrada (x), y por una transformación matemática se representa los datos de entrada de una manera diferente (y). [9]

Figura 3. Funcionamiento de una neurona [9]

En la Figura 4 se ve una red neuronal la cual es ocupada para trabajos más complejos, en donde se ocupan más neuronas teniendo como resultado una red neuronal en donde se ocupan varias entradas, transformaciones matemáticas y salidas. A medida que se tiene más neuronas es capaz de realizar tareas más complejas. [9]

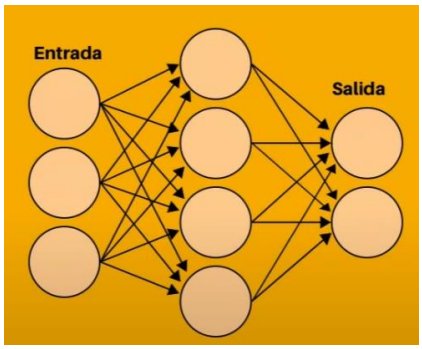


Figura 4. Red Neuronal [9]

d. Capacidad de computador para entrenamiento

Anteriormente se ocupaban CPU o unidades centrales de procesamiento, pero por el auge de los videojuegos los computadores de la actualidad vienen con GPU o unidades de procesamiento gráfico los cuales son mucho más potentes porque tienen más memoria y capacidad de cómputo. Recientemente GOOGLE creó un nuevo circuito integrado que se llama TPU, la cual es unidad de procesamiento de tensores, la cual está pensado únicamente para aplicaciones de Deep Learning, esta unidad de procesamiento es mejor que las GPU. [10]

e. Tipos de datos

- ***Estructurados:*** Tienen organización por ejemplo una tabla de datos o datos organizados a través de una gráfica. Obedecen algún tipo de organización

- ***No Estructurados:*** No tienen organización u orden aparente, como audio, texto, imágenes y videos.

1.3.2 Procesamiento digital de señales e imágenes

El procesamiento digital de imágenes es una rama que está presente en aplicaciones industriales, ganadería, agricultura, ciencias médicas, biometría, etc. Tiene como objetivo aplicar técnicas para facilitar el reconocimiento de determinados patrones y mejorar su calidad. Se usan herramientas del análisis complejo como transformada de Fourier, Transformada Z o Teorema de Convolución usados para crear filtros o tratamiento de señales. [11]

La Figura 5 muestra el sistema básico de DSP, en el cual entra una señal análoga como puede ser la voz, audio o video. Este ingresa a un filtro análogo y por medio de convertidores analógicos a digital (ADC) y convertidores digitales a analógicos (DAC), transforman la información de analógica a continua. [12]

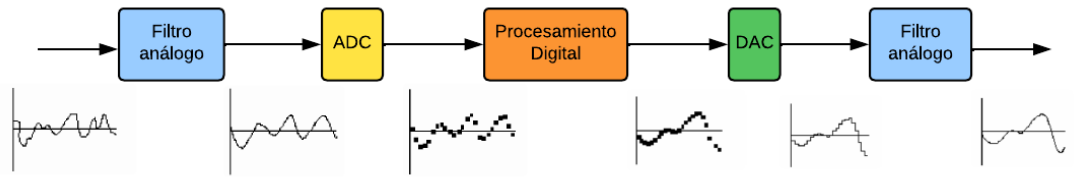


Figura 5. Sistema de DSP [12]

a. Procesamiento digital de imágenes

El procesamiento digital de imágenes engloba un conjunto de técnicas y métodos utilizados con el propósito de revelar o resaltar la información contenida en una

imagen. Este ámbito de estudio representa una destacada especialización en el campo informático y guarda estrecha relación con el procesamiento digital de señales. La evolución de las técnicas en este campo se origina en dos áreas principales de aplicación: la mejora de la información visual para la interpretación humana y el tratamiento de datos de la imagen destinado a la percepción de máquinas autónomas. Esto incluye fases como la transmisión y/o almacenamiento de los datos resultantes. En la actualidad, el procesamiento digital de imágenes desempeña un papel crucial en diversas disciplinas, contribuyendo significativamente a la mejora de la calidad visual y la interpretación automatizada de la información visual como se observa en la Figura 6. [13]

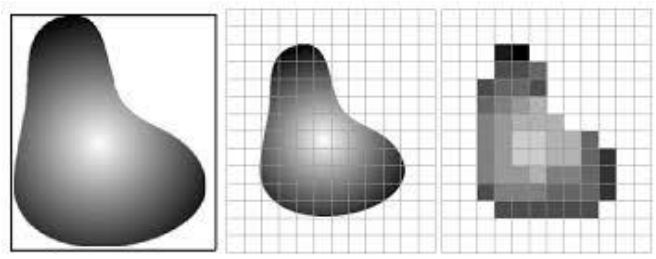


Figura 6. DSPI [13]

b. Características del procesamiento digital de imágenes

El procesamiento digital de imágenes representa una disciplina tecnológica de creciente importancia en la informática contemporánea, dedicada al análisis y manipulación de datos visuales.

Tabla 1. Características del procesamiento de imágenes [14]

Característica	**Descripción**
Mejora de la Calidad Visual	Aplicación de filtros y técnicas de mejora para optimizar la claridad, contraste y detalles de una imagen, facilitando la interpretación humana y el reconocimiento automático.
Segmentación de Imágenes	División de una imagen en segmentos o regiones para analizar y procesar por separado, permitiendo un enfoque más detallado en áreas específicas de interés.
Extracción de Características	Identificación y aislamiento de atributos específicos de una imagen, como bordes, colores dominantes o texturas, para un análisis más profundo y preciso.
Transformaciones Espaciales	Aplicación de operaciones geométricas para modificar la estructura espacial de la imagen, como rotaciones, escalas o traslaciones.
Filtrado de Imágenes	Utilización de filtros para suavizar o resaltar características en una imagen, contribuyendo a la reducción de ruido o la detección de bordes.

Su evolución ha sido significativa, abarcando desde la mejora de la interpretación visual humana hasta el desarrollo de capacidades avanzadas de percepción para sistemas autónomos. [14]

La Tabla 1. proporciona una visión general de algunas características esenciales del procesamiento digital de imágenes y destaca su diversidad de aplicaciones y beneficios en distintos contextos

c. Aplicaciones del procesamiento digital de imágenes

El uso creciente de tecnologías visuales ha posicionado al procesamiento digital de imágenes como un componente esencial en diversas áreas de la ciencia.La Tabla 2 proporciona una visión general de diversas aplicaciones del procesamiento digital de imágenes, evidenciando su diversidad y contribución a la resolución de desafíos en distintos dominios. [15]

Tabla 2. Aplicaciones del procesamiento digital de imágenes [15]

Aplicación	**Descripción**
Visión por Computadora	Implementación de algoritmos para permitir a las máquinas interpretar y comprender visualmente su entorno, con aplicaciones en robótica, vehículos autónomos y sistemas de vigilancia.
Análisis de Imágenes Satelitales	Procesamiento de imágenes satelitales para extraer información geoespacial, monitorear cambios ambientales, evaluar cultivos agrícolas y realizar análisis cartográficos detallados.
Automatización Industrial	Aplicación de técnicas de visión artificial para inspección automática de productos, control de calidad y optimización de procesos en entornos industriales, mejorando la eficiencia operativa.
Seguridad y Vigilancia	Utilización de cámaras y sistemas de vigilancia con procesamiento de imágenes para la detección de anomalías, seguimiento de objetos y garantizar la seguridad en entornos públicos y privados.
Realidad Aumentada y Virtual	Integración de imágenes procesadas para mejorar la experiencia visual en aplicaciones de realidad aumentada y virtual, permitiendo la superposición de información digital en entornos reales o virtuales.
Reconocimiento Automático de Objetos	Desarrollo de sistemas capaces de identificar y clasificar objetos en imágenes, con aplicaciones en reconocimiento facial, clasificación de productos y seguridad biométrica.

1.3.3 IA con Procesamiento digital de imágenes

La IA con procesamiento de señales e imágenes ha permitido generar mejoras significativas en muchos campos tales como procesar audio, tratamiento de imágenes entre otros. La IA ayuda a la automatización y optimización de toda tarea que sea analítica o compleja que en el pasado requería enfoques manuales. Entre las técnicas

de IA con DSP se encuentran las denominadas redes neuronales, algoritmos de aprendizaje. [16]

1.3.4 Adquisición de imágenes

La adquisición de imágenes por medio de procesamiento digital de imágenes es el proceso de capturar información visual del mundo real utilizando dispositivos electrónicos especializados, como cámaras o escáneres, y convertir esa información en datos digitales que pueden ser almacenados y procesados por computadoras. Estos datos digitales pueden ser sometidos a diversas técnicas de procesamiento de imágenes para mejorar su calidad, extraer características relevantes o realizar análisis cuantitativos, lo que permite una amplia gama de aplicaciones en campos como la medicina, la robótica, la seguridad y el entretenimiento. [17]

1.3.5 Visión artificial

La visión artificial por medio de procesamiento digital de imágenes es una disciplina que se enfoca en el desarrollo y aplicación de algoritmos y técnicas computacionales para permitir que las computadoras adquieran, analicen e interpreten imágenes y videos digitales. Su objetivo es dotar a las máquinas de la capacidad de comprender el contenido visual del mundo real, incluyendo la detección y reconocimiento de objetos, la extracción de características, el seguimiento de movimiento y la comprensión de escenas. La visión artificial encuentra aplicaciones en diversos campos, como la robótica, la medicina, la seguridad, el procesamiento de imágenes médicas y la conducción autónoma. [18]

1.3.6 Reconocimiento de patrones

El reconocimiento de patrones es una rama de la visión artificial que se centra en el desarrollo de algoritmos y técnicas para identificar y reconocer patrones en imágenes o datos visuales. Estos patrones pueden ser características específicas, como formas, texturas, colores o distribuciones espaciales, y el objetivo es extraer información relevante de los datos visuales para su análisis o toma de decisiones. El reconocimiento

de patrones tiene aplicaciones en diversos campos, como el reconocimiento facial, la detección de objetos, el análisis de imágenes médicas, la clasificación de documentos, entre otros. Los algoritmos utilizados en el reconocimiento de patrones incluyen técnicas de aprendizaje automático, como redes neuronales, máquinas de vectores de soporte, bosques aleatorios, entre otros. [19]

1.3.7 Microcomputador

Son chips en donde viene integrado una CPU, memoria y periféricos que tendría un ordenador, pero este está orientado a ejecutar tareas concretas y controlar varios sistemas electrónicos. [20]

a. NVIDIA Jetson Nano

Es una computadora enfocada en usos de Inteligencia Artificial en la cual se pueden cargar modelos de Deep o Machine Learning entrenados para que pueda ser usado en aplicaciones como la de visión artificial. Este modelo cuenta con varios puertos para poder ser ocupada como una PC. La clave para que esta computadora pueda ser ocupada en visión artificial es que cuenta con una GPU integrada lo cual la hace poderosa para proyectos de este tipo. Esta no viene con un sistema operativo por lo cual el usuario puede escoger y cargarla desde una memoria SD. Gracias a su arquitectura esta placa es una de las mejores para realizar aplicaciones de visión artificial, automatismo, domótica, etc. [20]

Figura 7. NVIDIA Jetson Nano [20]

b. Raspberry Pi 4

El Raspberry 4 es un SBC que tiene las capacidades para poder ser ocupada en la aplicación de Inteligencia Artificial. Esta cuenta con un procesador Broadcom BCM2711 de 64 bits. Tiene opciones de 2, 4 y 8 GB de memoria RAM. Cuenta con una GPU integrada lo que permite una aplicación óptima para visión artificial. Permite la instalación de un sistema operativo y tiene varios puertos como el de wifi, ethernet y Bluetooth. En general es una buena opción para aplicaciones de IA, reconocimiento de patrones y visión artificial por un precio no tan elevado. [21]

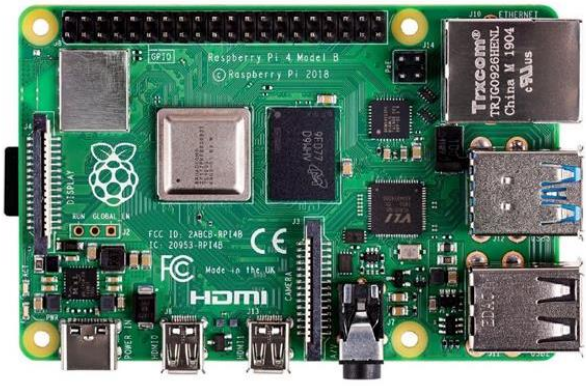


Figura 8. Rasberry pi 4 [21]

c. Coral Dev Board

Creado por Google es un dispositivo que está enfocado en la aplicación de modelos de aprendizaje automático. Este está creado netamente para modelos de inteligencia artificial pudiendo ejecutar modelos de Tensorflow y otros frameworks de aprendizaje automático. Dentro de sus características viene con un TPU, el cual es un procesador especializado en aprendizaje automático. Tiene 1gB de memoria RAM y tiene 8GB de memoria de almacenamiento incorporado, además de que incluye conectividad inalámbrica WiFi y Bluetooth. Es compatible con varios entornos de desarrollo, pero en especial los de Google. Es una buena opción en general para aplicaciones de IA además de tener buena eficiencia energética y capacidad de ejecutar modelos de alto rendimiento. [22]

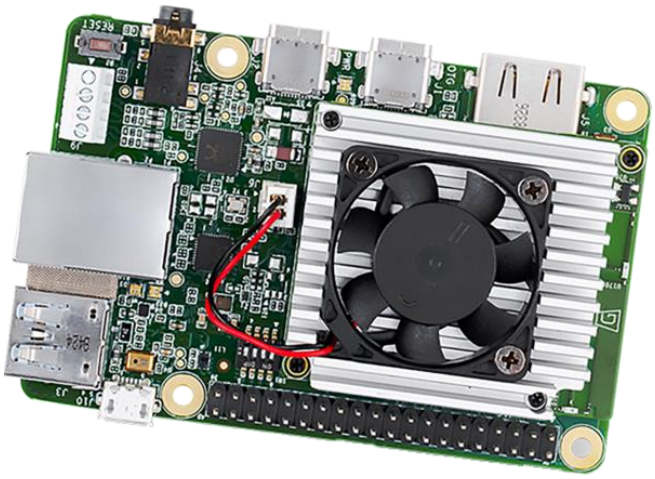


Figura 9. Coral Dev Board [22]

1.3.8 Plataformas basadas en la nube

Las plataformas basadas en la nube son servicios en línea que proporcionan acceso a recursos computacionales, almacenamiento de datos y herramientas de software a través de Internet. Estas plataformas permiten a los usuarios realizar tareas como el desarrollo de software, el análisis de datos y la ejecución de aplicaciones sin depender de recursos locales. En lugar de configurar y mantener sus propias infraestructuras, los usuarios pueden aprovechar los recursos informáticos, de almacenamiento y de servicios disponibles en la nube. [23]

a. Microsoft Azure Notebooks

Es una plataforma de computación en la nube desarrollada por Microsoft, que proporciona una interfaz basada en la web para la creación, ejecución y colaboración en cuadernos Jupyter. Estos cuadernos son entornos interactivos que permiten la combinación de código, texto y visualizaciones. Azure Notebooks es especialmente útil para proyectos de análisis de datos y desarrollo de modelos de aprendizaje automático, ya que ofrece integración con servicios en la nube de Microsoft y la posibilidad de acceder a recursos de cómputo escalables. [24]

b. Kaggle Kernels

Kaggle Kernels, por otro lado, es una función clave dentro de la plataforma Kaggle, diseñada para facilitar la colaboración en proyectos de ciencia de datos y aprendizaje

automático. Ofrece cuadernos Jupyter en la nube con acceso gratuito a unidades de procesamiento gráfico (GPU), lo que acelera el procesamiento de tareas intensivas en cómputo, como el entrenamiento de modelos. [25]

c. Google Colab

Google Colab, o Colaboratory, es una iniciativa de Google que proporciona un entorno de cuaderno Jupyter en la nube. Este servicio permite a los usuarios escribir y ejecutar código Python de forma gratuita, con la ventaja adicional de acceso a recursos de GPU. Colab es ampliamente utilizado en la comunidad de aprendizaje automático debido a su interfaz intuitiva y la capacidad de ejecutar código de manera eficiente en recursos de Google Cloud. [26]

1.3.9 Algoritmos de visión artificial

Un algoritmo de programación se define como un conjunto organizado de pasos o reglas que se implementan secuencialmente con el propósito de llevar a cabo una tarea o resolver un problema particular. Estos pasos se delinean de manera precisa y deben ser tanto finitos como ejecutables. Dada su naturaleza, los algoritmos son fundamentales en el ámbito de la programación, ya que ofrecen una estructura lógica crucial para abordar problemas y realizar diversas operaciones. [27]

a. YOLO

Es un popular algoritmo de detección de objetos en imágenes y videos. Fue propuesto por Joseph Redmon, Santosh Divvala, Ross Girshick y Ali Farhadi en 2016. La principal característica distintiva de YOLO es su capacidad para realizar detecciones de objetos en una sola pasada a través de la imagen, en lugar de requerir múltiples pasadas como algunos otros enfoques. Dentro de sus características tiene detección en una sola pasada, predicciones simultáneas, anclajes. Es utilizado ampliamente en aplicaciones de visión por computadora, como vigilancia, detectar objetos como vehículos, monitorizar el tráfico entre otros. [28]

b. SSD (Single Shot Multibox Detector)

SSD, o "Single Shot Multibox Detector," es un algoritmo diseñado para la detección de objetos en imágenes. Su característica distintiva es la capacidad de realizar detecciones en una única pasada a través de la imagen. Al dividir la imagen en una cuadrícula, SSD predice simultáneamente múltiples cajas delimitadoras y las probabilidades de pertenencia a diferentes clases para cada celda. Además, utiliza cajas delimitadoras de variados tamaños y relaciones de aspecto para abordar la detección de objetos en diferentes escalas y formas. [29]

c. Faster R-CNN (Region-based Convolutional Neural Network)

Faster R-CNN, cuyo nombre se deriva de "Region-based Convolutional Neural Network" (Red Neuronal Convolucional basada en Regiones), es un algoritmo sofisticado diseñado para la detección de objetos en imágenes. Su innovación principal radica en la incorporación de regiones de interés (RoI) para identificar áreas prometedoras en la imagen antes de llevar a cabo las predicciones finales. Utiliza una red de regiones de interés para proponer áreas destacadas, y luego se integra con un clasificador para realizar predicciones detalladas, logrando así un equilibrio eficaz entre precisión y eficiencia. [30]

1.3.10 Plataformas de etiquetado

En el ámbito de la segmentación, se dispone de una diversidad considerable de herramientas para llevar a cabo el etiquetado. Actualmente, existen opciones de variada índole; sin embargo, a continuación, se destacarán aquellas que gozan de mayor reconocimiento y uso en la práctica.

a. Labelme

Labelme constituye una herramienta de etiquetado de código abierto desarrollada en el lenguaje de programación Python, empleando la biblioteca Qt11 para su interfaz gráfica. Esta aplicación facilita la anotación manual de polígonos, que abarcan desde círculos y rectángulos hasta líneas, tiras de líneas y puntos, en imágenes destinadas a la detección, clasificación y segmentación de objetos. La parte más importante de

Labelme se encuentra en la inspiración tomada de la herramienta de anotación creada por el Laboratorio de Ciencias de la Computación e Inteligencia Artificial del Instituto Tecnológico de Massachusetts (CSAIL), conocida como LabelMe. Este último proyecto constituye un conjunto de datos dinámico, de acceso gratuito y abierto a contribuciones públicas mediante su propia herramienta de etiquetado. Los usuarios llevan a cabo modificaciones en las imágenes a través de esta plataforma hasta alcanzar su etiquetado preciso, y dichas alteraciones se almacenan y están disponibles públicamente para su descarga desde el repositorio de datos de LabelMe. Este enfoque dinámico implica que los datos experimentan cambios continuos debido a las contribuciones de la comunidad de usuarios que hacen uso de la herramienta. [31]

Cabe señalar que, en Labelme, solo es factible guardar las etiquetas en formato JSON directamente desde la aplicación; no obstante, en el repositorio de LabelMe se encuentra disponible un script en Python que permite la conversión de estas anotaciones al formato PASCAL VOL. No se ofrece compatibilidad con otros formatos, como YOLO y COCO, en esta aplicación específica. [31]

Figura 10. Interfaz de Labelme

b. Keylabs

KeyLabs se presenta como una plataforma de etiquetado de datos, esta vez sin disponibilidad de código abierto, diseñada por expertos en anotación con el propósito de brindar funcionalidades de alto rendimiento y sistemas de gestión distintivos. Esta aplicación ofrece la posibilidad de asignar roles de usuario y permisos específicos para cada proyecto individual, tales como anotador, administrador de proyectos,

superusuario y administrador. Permite la marcación de objetos que se encuentran fuera del cuadro de video durante un período de tiempo, capturándolos cuando vuelven a aparecer mediante las funciones de línea de tiempo de objetos. Admite la creación de metadatos, incluyendo estructuras jerárquicas; por ejemplo, los objetos pueden establecer relaciones "padre/hijo" entre sí. Posibilita la extracción de etiquetas en formato JSON. [32]

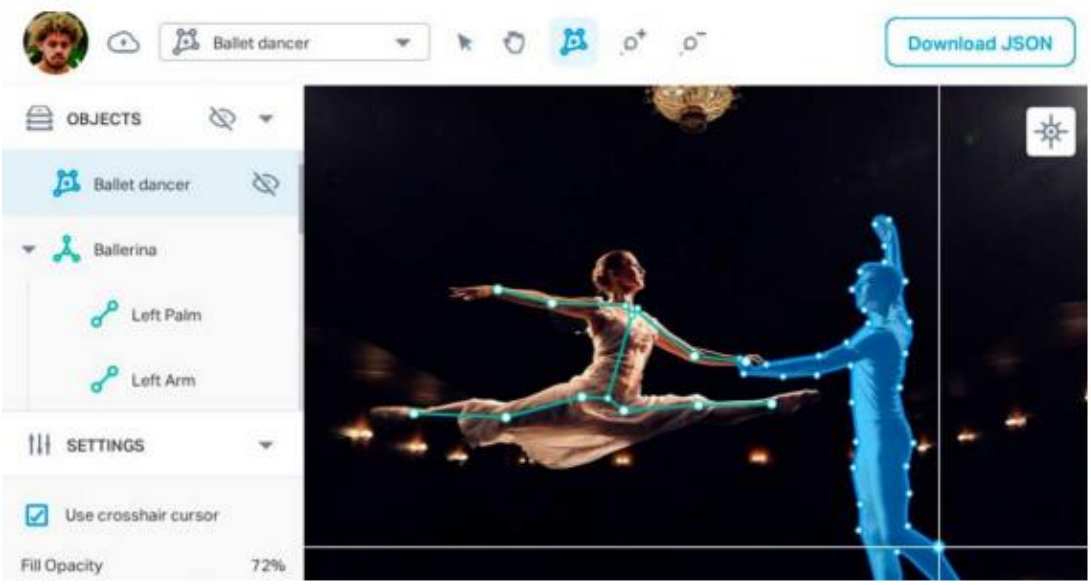


Figura 11. Interfaz de Keylabs [32]

c. CVAT

La Herramienta de Etiquetado de Objetos Visuales (VoTT) se configura como una aplicación de carácter gratuito y de código abierto destinada a la anotación y etiquetado de imágenes, siendo desarrollada por Microsoft. Este software, creado en TypeScript, tiene como finalidad la generación de modelos integrales de detección de objetos a partir de activos de imágenes y videos, empleados posteriormente en algoritmos de visión por computadora. [33]

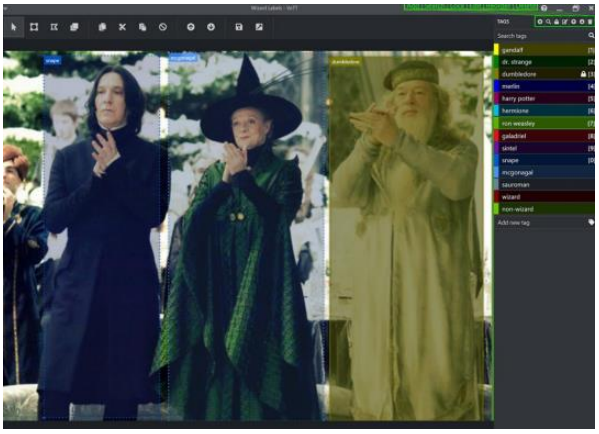


Figura 12. Interfaz de VoTT [33]

1.3.11 Tecnologías de Seguridad

Las tecnologías de seguridad se refieren al conjunto de herramientas, dispositivos y sistemas electrónicos diseñados para detectar, alertar y supervisar eventos o actividades que representen un riesgo para la seguridad. Estas tecnologías pueden incluir sistemas de alarmas audibles, sistemas de videovigilancia con cámaras, sensores de movimiento, sistemas de control de acceso y otros dispositivos que funcionan de manera integrada para proteger personas, propiedades y activos. Estas tecnologías proporcionan una capa adicional de protección, permitiendo una respuesta rápida y la toma de acciones para prevenir incidentes y salvaguardar la seguridad en diversos entornos, como residencias, empresas, instituciones y espacios públicos. [34]

1.3.12 Sistemas de seguridad

Estos sistemas utilizan sensores, cámaras, dispositivos de detección y sistemas de comunicación para recopilar información en tiempo real y generar alertas o notificaciones cuando se detectan anomalías o violaciones de seguridad. El objetivo principal de estos sistemas es proporcionar una respuesta rápida y eficaz ante situaciones críticas, permitiendo la toma de decisiones oportunas y la implementación de medidas correctivas para proteger personas, propiedades y activos. [35]

1.3.13 Sistemas de seguridad en locales comerciales

En los locales comerciales siempre existen 2 sistemas esenciales dependiendo de la seguridad del local. Sistemas de video vigilancia, sistemas de alarmas. Para una mejor compresión de cómo están conformados estos sistemas se lo puede apreciar en la en Tabla 3. [36]

Tabla 3. Sistemas de seguridad [36]

SISTEMAS	DETALLE
Sistemas de videovigilancia	-Cámaras de seguridad que graban y monitorean actividades dentro y alrededor del local. -Sistemas de CCTV (Circuito Cerrado de Televisión) para la supervisión en tiempo real. -Grabadores de video que almacenan las imágenes para su revisión posterior.
Sistemas de Alarmas	-Alarmas contra intrusos que se activan cuando se detecta movimiento no autorizado. -Sensores de puertas y ventanas que alertan sobre intentos de acceso no autorizado. -Sistemas de alarma conectados a centrales de monitoreo para una respuesta rápida.

1.3.14 Análisis de los sistemas de videovigilancia

Para conocer los precios de un sistema de videovigilancia se realizó una proforma con INNOVA ILUMINACION una empresa de Ambato. Se sacaron los posibles valores más la mano de obra de sus proformas.

a. DVR

Un Digital Video Recorder, es un dispositivo electrónico diseñado para capturar, almacenar y gestionar señales de video provenientes de cámaras de seguridad. Este dispositivo, fundamental en sistemas de videovigilancia, permite la grabación y reproducción de imágenes, facilitando así la monitorización y análisis de eventos ocurridos en tiempo real o en momentos específicos. La función principal del DVR es procesar las señales analógicas de las cámaras de vigilancia, convirtiéndolas en formato digital para su almacenamiento eficiente en medios de almacenamiento, como

discos duros. Además, ofrece opciones de búsqueda y reproducción, facilitando la revisión de grabaciones según las necesidades de seguridad. Un DVR de 4 canales puede gestionar la entrada de video de hasta 4 cámaras de vigilancia al mismo tiempo. Esto es particularmente útil en sistemas de seguridad donde se requiere monitorear múltiples puntos de interés. [37]

A continuación, se puede observar algunas características y precios de algunos de ellos que se encuentran dentro del mercado a nivel nacional dentro de la Tabla 4.

Tabla 4. Especificaciones de DVR [37]

DVR	ESPECIFICACIONES	PRECIO
DVR HIKVISION 4 CH TRUBO HD 1080P	- Desarrollado por tecnología eSSD - Disco Duro eSSD de 300GB (Graba Aproximadamente 2 Semanas todos los Canales) - Bajo consumo de energía	$145,00
DVR 8 CANALES HIKVISION ACUSENSE	- DVR de 8 canales Turbo HD 5.0 a 8MP - Análogo/IP Hasta 12 canales IP de hasta 8mp - Salida VGA/HDMI 4K	$260,00
DVR 4 CANALES HIKVISION ACUSENSE 5MPX	- Dispone de 4 canales BNC de 75 Ohm – Soporte en cámaras HDTVI/AHD/CVI/CVBS/IP video inputs – Resolución de grabación 8MP, 4 MP, 1080p Lite, 720p, WD1, 4CIF, VGA, CIF	$159,00

b. Cámaras

La eficacia de los sistemas de videovigilancia se encuentra estrechamente ligada a diversos factores, entre los que destacan la calidad de resolución, las características técnicas, el costo y las funciones programadas en las cámaras utilizadas. Estos aspectos serán analizados detalladamente en secciones posteriores de la investigación para ofrecer una visión más exhaustiva. [38]

En términos de costos, se observa un promedio aproximado de alrededor de 60 dólares por cada cámara del sistema. Sin embargo, es crucial tener en cuenta que estas cámaras están diseñadas principalmente para funciones de monitoreo. [38]

c. Materiales básicos para la instalación de cámaras

En la Tabla 5 se puede apreciar un promedio de los precios de los materiales básicos que se necesitarían para un sistema de vigilancia. [38]

Tabla 5. Materiales básicos para un sistema de vigilancia. [38]

Cantidad	Descripción	V. UNITARIO	V. TOTAL
2	CAJETIN SOBREPUESTO BLANCO	1,85	3,70
1	CÁMARA	60	60
2	TAPAS CIEGA	0,75	1,50
1	CABLE UTP POR METROS	0,45	0,45
2	BALUMS	6,50	13,00
2	FUENTES	4,00	8,00
1	DVR HIKVISION 4 CH TRUBO HD 1080P	145,00	145,00
1	TAYPE, TORNILLOS	2,00	2,00
1	INSTALACION Y MANO DE OBRA	45,00	45,00
		TOTAL	$278

1.3.15 Sistemas de alarmas

La Tabla 6 ofrece una visión resumida de tres sistemas de alarmas, con la intención de proporcionar información clave para la toma de decisiones. Se ha estimado el precio de cada sistema en dólares, considerando una variedad de características como tipo de alarma, zonas de cobertura, sensores, control remoto, monitoreo 24/7, cámaras de seguridad, autonomía de batería, notificaciones y otros aspectos relevantes. Es importante tener en cuenta que los precios son aproximados y pueden variar según la marca, modelo y funciones adicionales incluidas en cada sistema de alarma. Esta comparación pretende ser una guía informativa para aquellos que buscan evaluar diferentes opciones de sistemas de seguridad. [39]

Tabla 6. Sistemas de alarmas [39]

Características	**Sistema A**	**Sistema B**	**Sistema C**
Tipo	Alarma inalámbrica	Alarma cableada	Sistema híbrido (cableado e inalámbrico)
Sensores	Infrarrojos, magnéticos	Movimiento y contactos	Infrarrojos y de contacto
Monitoreo 24/7	Opcional con suscripción	No disponible	Incluido sin costo adicional
Cámaras de Seguridad	No incluidas	No incluidas	Cámaras IP incluidas
Notificaciones	Alertas por aplicación	Llamada telefónica	Alertas por aplicación y mensaje de texto
Precio Estimado	$300 - $400	$200 - $250	$450 - $550

1.3.16 Delitos en locales comerciales

Los negocios en Ecuador enfrentan diversos desafíos en términos de seguridad, con el problema persistente de delitos en locales comerciales. Entre las preocupaciones comunes se encuentran los robos y hurtos, donde los delincuentes buscan efectivo y productos valiosos. Los asaltos a mano armada representan una amenaza seria, con consecuencias potencialmente graves para empleados y clientes. Además, los delitos financieros, como el fraude con tarjetas, pueden afectar la estabilidad económica de los negocios. El vandalismo también es una problemática, con actos que van desde grafitis hasta daños a la propiedad. La seguridad de los empleados es una prioridad, ya que pueden estar en riesgo durante eventos delictivos. Para abordar estos problemas, se implementan medidas como sistemas de vigilancia, alarmas y políticas de seguridad, y la colaboración con las autoridades y la participación en programas de prevención son estrategias clave. [40]

1.3.17 Tipos de robos

En Ecuador, diversas modalidades de robos impactan a la sociedad, siendo el robo común o hurto una forma en la que los delincuentes se apropian de propiedad sin violencia directa, incluyendo la sustracción de objetos de valor. Los robos a mano armada presentan un riesgo significativo al involucrar armas de fuego o blancas para amenazar a las víctimas. El robo de vehículos, el robo a domicilios y el robo de identidad son problemáticas adicionales, afectando tanto a la propiedad como a la seguridad personal. El fraude empresarial y el robo de mercancía en comercios también son preocupaciones, con empleados deshonestos y ladrones que buscan obtener ganancias ilícitas. La prevención del delito y la seguridad ciudadana son fundamentales, involucrando estrategias gubernamentales, medidas de seguridad privada y la participación activa de la comunidad. La situación delictiva puede evolucionar, por lo que se recomienda consultar fuentes locales actualizadas para obtener información más reciente sobre la situación en Ecuador. [40]

1.4 Objetivos

1.4.1 Objetivo general

- Desarrollar un sistema de detección de situaciones delictivas en establecimientos comerciales usando inteligencia artificial.

1.4.2 Objetivos específicos

- Analizar las características de los sistemas de seguridad utilizados en locales comerciales.
- Seleccionar un algoritmo de inteligencia artificial para el análisis de video que detecte situaciones delictivas.
- Establecer los elementos de hardware y software que componen el sistema de detección automática.
- Evaluar el algoritmo implementado para la detección de actividad delictiva.

CAPÍTULO II. METODOLOGÍA

2.1 Materiales

En el marco del proyecto, se ha concebido un sistema de detección de delitos en locales comerciales por medio de inteligencia artificial, fundamentado en el lenguaje de programación Python. Se ocupó la arquitectura YOLO para la detección de imágenes, mientras que, para llevar a cabo el etiquetamiento, se utilizó la herramienta Labelme. El proceso de entrenamiento se llevó a cabo empleando un conjunto de datos propio que consta de 2000 imágenes. Para el procesamiento eficiente de los datos, se emplearon una NVIDIA Jetson Nano y una cámara IP. Con el propósito de transmitir datos de manera efectiva, se empleó una plataforma de mensajería instantánea conocida como Telegram. Esta plataforma se encarga de enviar información de ubicación y capturas en situaciones donde se detecte alguna anomalía. En las Tabla 7 y Tabla 8 se describen el hardware y software que se ocupara en el sistema.

Tabla 7. Características del hardware seleccionado

Material	**Características**	**Descripción**
NVIDIA JETSON NANO	Microordenador dedicado a IA	Microcomputador que integra un potente procesador y una GPU para ejecutar cargas de trabajo de IA en tiempo real. Diseñada para ser compacta y eficiente.
Ups Apc Easy Bv800	450W y 6 tomas	Es un sistema de alimentación ininterrumpida diseñado para proteger equipos electrónicos como computadoras, enrutadores y módems de cortes de energía.
Rohxizw 1520p Hd Wifi	1520p, Wifi, visión 360	Cámara de seguridad con buena resolución, visión nocturna, se conecta a WiFi y permite audio bidireccional.

Tabla 8. Características del software seleccionado

Material	**Características**	**Descripción**
Yolov8m	Arquitectura de vision artificial	Se basa en una arquitectura de red neuronal profunda con mejoras significativas en la extracción de características y la eficiencia del modelo.
Python	Lenguaje de programación	Es un lenguaje de programación de alto nivel, interpretado, de propósito general y multiparadigma.
FireBase	Base de datos de tipo NoSQL	Es una plataforma en la nube desarrollada por Google que facilita el desarrollo de aplicaciones web y móviles de alta calidad.
NGROK	Herramienta de desarrollo y túnel seguro	Es una herramienta que permite exponer un entorno local (localhost) a internet, de forma segura y sin necesidad de configuraciones complejas.

2.2 Métodos

2.2.1 Modalidad de la investigación

A continuación. Se describen los diferentes tipos de investigación que se aplicaron para la ejecución del proyecto de investigación.

El presente proyecto fue una investigación aplicada, porque fueron aplicados los conocimientos ya existentes para poder entrenar una red neuronal que detecte situaciones delictivas, se empleó conocimientos tanto teóricos como prácticos para tener un buen prototipo.

El trabajo de investigación también contó con investigación bibliográfica, porque se sustenta mediante la recopilación de información de revistas técnicas, libros, artículos científicos, publicaciones de internet, bootcamps, tesis que tengan relación con la visión artificial, actividades delictivas y más herramientas útiles que se usaron el proceso.

El proyecto fue una investigación de campo, debido a que se recopiló información, se grabó videos para segmentar y obtener una base de datos eficiente y se implementó el sistema donde se origina el problema, es decir en un local comercial donde ya ha sucedido incidentes.

2.2.2 Recolección de información

Para la recolección de información se usaron libros, proyectos, artículos científicos de donde se extrajo la información y ejemplos de cómo aplicar un modelo de inteligencia artificial. Para la creación de los dataset a ser usados en el entrenamiento, se obtienen de repositorios en confiables disponibles en la web y se creó un nuevo repositorio de imágenes adquiridas en el local comercial donde se implementó el sistema.

2.2.3 Procesamiento y análisis de datos

Para el procesamientos y análisis de datos se realizaron los siguientes pasos:

• Búsqueda de información referente al tema.

• Selección de fuentes adecuadas y útiles.

• Observación y estudio de la metodología usada para una óptima elaboración de sistemas de monitoreo.

• Discerniendo la información más adecuada para dar la solución al planteamiento del problema.

CAPÍTULO III. RESULTADOS Y DISCUSIÓN

3.1 Desarrollo de la propuesta

Para la ejecución del proyecto, se inició con la identificación de elementos que pueden ser identificados en video vigilancia en situaciones delictivas en locales comerciales. A continuación, se establecen los elementos físicos y de software necesarios en el sistema. Para el entrenamiento del sistema de detección de situaciones delictivas se seleccionó el modelo Yolov8. El proceso de etiquetado de datos se realizó utilizando la herramienta LabelMe, donde se etiquetaron los DataSet. El entrenamiento se realizó en la plataforma Google Colab con Yolov8 Ultralitycs.

3.1.1 Definición de los elementos para la propuesta

Se realizó un análisis estadístico de la criminalidad en Ecuador, destacando datos reveladores sobre la distribución geográfica y las tasas de homicidios. Según las estadísticas recopiladas hasta el 5 de diciembre de 2023, se observa que 10 de las 24 provincias concentran más del 90% de los crímenes en el país [41].

La cifra total de muertes violentas, que incluye asesinatos, homicidios, sicariatos y femicidios, asciende a 7.258 hasta la fecha mencionada. La alarma se intensifica al constatar que ha ocurrido, en promedio, un asesinato por hora en Ecuador [41].

Este nivel de violencia no solo se limita a enfrentamientos entre grupos criminales, sino que también engloba efectos como: robos, asaltos, secuestros y extorsiones lo que ocasiona víctimas colaterales.

A finales de 2023, Ecuador como catalogado como el país más inseguro de Latinoamérica, teniendo una tasa de muertes violentas superior a los 40 por cada 100.000 habitantes. [42]

d. Elementos a detectar en situaciones delictivas

Para el desarrollo de este proyecto se han escogido los elementos más comunes que existen en un asalto como: armas de fuego, armas blancas, la presencia de personas

con pasamontañas y el gesto de peligro más común cuando una persona al sentirse amenazada. En la Tabla 9 se describen las características de los elementos a detectar.

Tabla 9. Elementos comunes en un robo [43]

Elemento	Figura	Características
Pistolas (Arma de Fuego)		Las armas de fuego son dispositivos diseñados para lanzar proyectiles utilizando la energía liberada por la combustión de un propelente, generalmente pólvora. Dentro de estas tenemos pistolas, revólveres, fusiles, escopetas, etc.
Cuchillos (Armas Blancas)		Un arma blanca es un tipo de arma cuya principal característica es que su capacidad de causar daño se debe al filo o a la punta afilada de su hoja. Dentro de estos se encuentran cuchillos, dagas, machetes,etc.
Pasamontañas		Es una prenda de vestir diseñada para cubrir la cabeza y, en ocasiones, el cuello y la cara, dejando solo al descubierto ciertas partes como los ojos, la boca o, en algunos casos, solo los ojos.
Gestos de peligro		Levantar las manos puede ser percibido como un gesto para demostrar que la persona no representa una amenaza y que no tiene la intención de resistirse al robo.

3.1.2 Tecnologías utilizadas

El análisis no solo se centra en las características técnicas de cada componente, sino también en la evaluación minuciosa de los costos asociados. La combinación estratégica de estas herramientas y dispositivos establece una sólida base para el éxito en la implementación del proyecto, optimizando tanto la funcionalidad técnica como la eficiencia económica del sistema propuesto. Además de buscar la arquitectura que mejor se adapte al sistema.

a. Selección de microcomputador

Para el correcto funcionamiento del sistema de visión artificial es necesario seleccionar un microcomputador con las características adecuadas que permita el procesamiento de video a una apropiada velocidad, teniendo en cuenta que el sistema de detección de situaciones delictivas se ejecuta en tiempo real. En la Tabla 10 se realiza la comparación de las principales de los principales microcomputadores.

Tabla 10. Comparación de características principales de los microcomputadores [44], [45], [46].

TIPO	**RASPBERRY PI 4**	**NVIDIA JETSON NANO**	**CORAL DEV BOARD**
GPU	VideoCore VI 500 MHz	Maxwell de 128 núcleos	Google Edge TPU
CPU	Broadcom BCM2711 de cuatro núcleos Cortex-A72 de 64 bits a 1.5 GHz.	Quad-core ARM Cortex-A57 de 64 bits.	Quad-core ARM Cortex-A53 de 1.5 GHz.
Puertos	Micro-HDMI, USB, Ethernet, tarjeta microSD, alimentación USB-C. audio y video, pines GPIO.	Conector Gigabit Ethernet, 4 puertos USB 3.0 y un puerto para cámara MIPI CSI-2.	USB 3.0, HDMI, un conector de cámara MIPI-CSI, un conector de pantalla, un conector de audio, un conector Ethernet, y un conector microSD.
Memoria	2Gb,4Gb y 8Gb	4 GB de memoria RAM	8 GB de memoria RAM LPDDR4.
Conectividad	Wi-Fi 802.11ac de doble banda y Bluetooth 5.0. También tiene puertos para Gigabit Ethernet	Wi-Fi de doble banda (2.4 GHz y 5 GHz), ethernet, Bluetooth.	Wi-Fi 802.11b/g/n/ac y Bluetooth 4.1.
Almacenamiento	microSD (no incluye)	microSD (no incluye)	Cuenta con 8 GB de almacenamiento eMMC incorporado.
Precio	$ 260	$ 200	$ 240

Debido a su alto rendimiento y utilidad en proyectos de inteligencia artificial que necesiten aprendizaje profundo se seleccionó la NVIDIA Jetson Nano. Por otro lado, el Raspberry Pi 4 tiene capacidades para proyectos de IA más pequeños para hacer clasificaciones de imágenes o patrones más básicos. Por último, la Coral Dev Board es un microcomputador potente debido a que esta tiene una TPU y es muy buena en términos de aplicaciones de visión artificial. Teniendo en cuenta el precio y funcionalidad se optó por la NVIDIA Jetson Nano dado a que por su GPU tiene una

buena capacidad de procesar modelos de IA, teniendo buen rendimiento en general para proyectos con visión artificial.

b. Selección de la cámara de videovigilancia

En la Tabla 11 se realiza una comparación de las características de las cámaras que puedan ser utilizadas en la vigilancia de locales comerciales.

Tabla 11. Comparación de características de las cámaras [47], [48], [49].

Característica	**Raspberry Pi Camera Module V2**	**Cámara celular- Xiaomi Redmi Note 8**	**Rohxizw 1520p Hd Wifi**
Resolución	8 MP	64 MP (3280 x 2464 píxeles)	1520p
Tipo de Conexión	CSI (15 pines)	USB- WIFI	Wifi
Micrófono Incorporado	No (Requiere configuración adicional)	Sí	Sí
Enfoque Ajustable	Sí	Sí	Sí
Tamaño Físico	Pequeño y compacto	Compacto	Compacto
Sistema Operativo Compatible	Raspberry Pi OS	Android	Windows, macOS, Linux
Precio	$ 25.00	$140.00	$20.00

Analizando las características se escogió la cámara Rohxizw, ya que tiene más ventajas debido con la compatibilidad, calidad en la imagen, resolución, giro 360, cabe recalcar que para obtener más precisión del funcionamiento del prototipo es recomendable entrenar con la cámara que se va a implementar para tener mejor precisión de detección.

c. Selección del UPS

Los Sistemas de Alimentación Ininterrumpida, comúnmente conocidos como UPS, desempeñan un papel crucial en la protección de dispositivos electrónicos sensibles ante diversos problemas relacionados con la energía eléctrica. Entre estos dispositivos se encuentran computadoras, servidores, sistemas de red, equipos médicos, sistemas de comunicación, y muchos otros. Estos dispositivos no solo cumplen la función de proporcionar energía continua en casos de apagones, sino que también desempeñan un papel esencial en salvaguardar los dispositivos contra posibles daños ocasionados por fenómenos como picos de voltaje, fluctuaciones en la energía y otros problemas asociados con la calidad de la energía eléctrica. El UPS actúa como una barrera

protectora al suministrar energía de respaldo inmediata cuando se interrumpe la alimentación principal. Además, contribuye a mantener la estabilidad de la tensión eléctrica, mediante la incorporación de un regulador de voltaje en algunos modelos, lo que protege aún más los dispositivos conectados. A continuación, en Tabla 12 se puede ver una tabla comparativa de las diferentes características de un UPS.

Tabla 12. Comparación de los UPS [50], [51], [52].

Características	**Ups Forza Nt-511**	**Ups Apc Easy Bv800**	**Apc Ups Apc Easy Bv650**
Capacidad W	250W	450W	375W
Numero de salidas	6 tomas	6 tomas	4 tomas
Precio	$51.00	$84.00	$74.99

Para el uso del proyecto se optó por usar la UPS de 450 W teniendo en cuenta el precio, la capacidad de batería que posee y el número de tomas siendo estas características importantes si se quiere escalar el proyecto.

d. Periféricos adicionales

Mouse: Dispositivo de entrada que se ocupa para interactuar con la interfaz gráfica del usuario (GUI). Es un sensor que detecta el movimiento al arrastrarlo en una superficie y tiene dos botones para realizar clics. Este facilita la manipulación de objetos en la pantalla con movimientos físicos. [53]

Teclado: Dispositivo que consiste en un conjunto de teclas con una tipografía alfanumérica para poder introducir datos o comandos, dentro de este se tiene letras, números, símbolos y varias funciones especiales. [54]

Monitor: Dispositivo que muestra información visual del dispositivo electrónico, este es un componente fundamental para la representación de los datos que se tiene. Este tiene varias tecnologías como LED, LCD u OLED para producir imágenes de calidad. [55]

e. Análisis de respaldo energético

Se seleccionó un UPS de 450 W, el cual suministra energía a la NVIDIA JETSON NANO, el celular y el Router que se encuentran en el lugar de aplicación del sistema. A continuación, se analiza la duración del suministro energético de respaldo, para lo cual se calcula el consumo energético de los tres dispositivos en una hora. En la Tabla 13 se presenta el consumo energético desglosado por componente. Para calcular el tiempo de respaldo que se tiene se lo hace por medio de la Tabla 13.

Tabla 13. Consumo energético

MATERIALES	**CONSUMO EN UNA HORA**
Router	5W/h
Nvidia jetson nano	10W/h
Celular	15W/h
TOTAL	30W/h

$$Potencia_{total\ consumida\ en\ una\ hora} = 30Wh$$

$$Potencia_{total\ suministrada} = 450W$$

$$\text{Horas}_{\text{respaldo}} = \frac{\text{Potencia}_{\text{total suministrada}}}{\text{Potencia}_{\text{total consumida en una hora}}} \quad (1)$$

$$Horas_{respaldo} = \frac{450W}{30W/h}$$

$$Horas_{respaldo} = 15h$$

a. Selección del software de programación

Para la selección del software de programación se realizó una comparación de las características de los lenguajes de programación más utilizados para poder realizar aplicaciones con IA. En la Tabla 14 que expone la comparación de características entre estos lenguajes, proporciona una herramienta esencial para evaluar sus capacidades y decidir cuál se ajusta mejor a las necesidades específicas del proyecto. Este análisis riguroso garantiza una elección informada y eficiente en la implementación de la programación necesaria para la integración de la inteligencia artificial en el sistema.

Tabla 14. Características de los lenguajes de programación [56], [57], [58].

Característica	MATLAB	Python	Lisp
Sintaxis	Sintaxis matricial, fácil para operaciones matemáticas.	Sintaxis clara y concisa, fácil de aprender.	Sintaxis basada en paréntesis, puede ser más compleja.
Bibliotecas/Frameworks	Amplias herramientas de procesamiento de señales, redes neuronales, y aprendizaje automático.	Abundantes bibliotecas como TensorFlow, PyTorch, scikit-learn.	Menos bibliotecas específicas para-IA, pero puede integrarse con algunas.
Desarrollo Rápido	Bueno para el desarrollo rápido de prototipos debido a su amplia variedad de herramientas.	Excelente para un desarrollo rápido, con sintaxis fácil de entender.	Puede tener un tiempo de desarrollo más largo debido a la sintaxis y herramientas específicas.
Aplicaciones Comunes	Procesamiento de señales, visión por computadora, simulación.	Visión por computadora, procesamiento de lenguaje natural, aprendizaje profundo.	Procesamiento de lenguaje natural, sistemas expertos.
Integración de Sistemas	Puede integrarse con sistemas embebidos y hardware específico.	Ampliamente utilizado en sistemas y aplicaciones empresariales.	Menos común en sistemas embebidos y empresariales.

Al final de la comparativa se optó por Python para el desarrollo del proyecto ya que se basa en su rica colección de librerías, una comunidad activa y su facilidad de uso. Estos factores hacen que Python sea una herramienta poderosa y accesible además de gratis para realizar proyectos que conlleven IA.

b. Selección de la plataforma basada en la nube

En el presente proyecto se requiere una plataforma basada en la nube para realizar el entrenamiento del modelo de inteligencia artificial para la detección de situaciones delictivas. Una plataforma basada en la nube se refiere a un conjunto de servicios y recursos informáticos que están disponibles a través de internet. En lugar de depender de recursos locales, como servidores físicos, esta plataforma utiliza la infraestructura de la nube para ofrecer acceso a aplicaciones, almacenamiento y otros servicios, facilitando la flexibilidad y el acceso remoto a través de la conectividad en línea. En la

Tabla 15, se tiene un análisis de cada una de las plataformas basadas en nube dependiendo de sus características.

Tabla 15. Plataformas basadas en la nube [59], [60], [61].

CARACTERÍSTICA	MICROSOFT AZURE NOTEBOOKS	KAGGLE KERNELS	GOOGLE COLAB
Entorno de Desarrollo	Jupyter en la nube	Jupyter en la nube	Jupyter en la nube
Acceso Gratuito a GPU	Sí	Sí	Sí
Integración con la Nube	Integración con servicios en la nube de Azure	Integración con servicios de Kaggle	Integración con Google Cloud Platform
Colaboración en Tiempo Real	Sí	Sí	Sí
Acceso a Recursos Escalables	Sí	Sí	Sí (a través de Google Cloud)
Almacenamiento y Carga de Datos	Integrado con Azure Storage	Integrado con Kaggle Datasets	Integrado con Google Drive
Herramientas Adicionales	Servicios adicionales de Azure	Funciones específicas de Kaggle	Acceso a servicios de Google Cloud
Integración con Herramientas ML	Integración con servicios de Azure ML	Funcionalidades específicas de Kaggle ML	Integración con TensorFlow, PyTorch, etc.

Una vez analizado todos los parámetros se optó por Google Colab debido a su acceso gratuito a recursos de GPU lo cual es de vital importancia al momento de realizar el entrenamiento de inteligencia artificial, integración sencilla con Google Drive en donde está la basa de datos, entorno basado en Jupyter Notebooks, preinstalación de bibliotecas populares de Python, facilidad de colaboración en tiempo real y acceso rápido a datos y conjuntos de datos públicos.

c. Selección de la arquitectura de visión artificial

La arquitectura para visión artificial se refiere al diseño estructurado de redes neuronales convolucionales y modelos de aprendizaje profundo diseñados específicamente para tareas relacionadas con el procesamiento de imágenes y la percepción visual. Estas arquitecturas son fundamentales en aplicaciones como la detección de objetos, reconocimiento facial, clasificación de imágenes, entre otras. En la Tabla 16 se puede observar una comparativa de las arquitecturas más usadas.

Tabla 16. Arquitecturas de visión artificial [62] , [63] , [64].

Criterio	YOLO	Faster R-CNN	SSD
Método de detección	Detección en una pasada	Proposición de regiones	Detección en una pasada
Precisión	Buena	Muy precisa	Buena
Velocidad de detección	Rápida	Más lenta que YOLO	Rápida
Tamaño de modelo	Moderadamente grande	Grande	Moderadamente grande
Uso en tiempo real	Sí, eficiente	No tan eficiente	No tan eficiente

Después de analizar y comparar distintas opciones, se eligió YOLO en sus versiones más recientes. Su arquitectura robusta y eficiente resulta beneficiosa para el entorno específico en el que se requiere un rendimiento óptimo, ya que se busca lograr una detección precisa con una velocidad adecuada para su uso en tiempo real.

d. Selección de la versión de Yolo

Para la selección del modelo hay que tener en cuenta las características que muestran, los cuales se describen a continuación:

Tabla 17. Características de las versiones de Yolov8 [65]

Parámetro	Descripción
Size en píxeles	Representa el tamaño de la imagen de entrada en píxeles que el modelo YOLO utiliza para realizar las detecciones. Por ejemplo, un tamaño de 416x416 significa que el modelo espera imágenes de 416 píxeles de ancho y 416 píxeles de alto.
MAP val 50-95	La columna "MAP val 50-95" generalmente indica el promedio de la precisión (precision) promedio para umbrales de confianza del 50% al 95%.
Speed CPU ONNX ms	Es un indicador del tiempo que tarda el modelo en procesar una imagen en una CPU.
Params M (Millones)	Indica la cantidad de parámetros en millones que tiene el modelo. Los parámetros son los pesos y sesgos que el modelo aprende durante el entrenamiento. Modelos más grandes pueden tener más parámetros.
FLOPs (Operaciones de punto flotante por segundo):	Representa la cantidad de operaciones de punto flotante por segundo que el modelo puede realizar durante la inferencia. Es una medida del rendimiento computacional del modelo. Modelos con más FLOPs pueden realizar más cálculos en menos tiempo.

Se eligió el modelo YOLOv8m debido a su destacada velocidad y precisión, comparándolo con otros modelos disponibles y teniendo en cuenta el parámetro de velocidad de la CPU, que indica la rapidez del sistema, y el MAP (mean average precision), que indica la efectividad del sistema. La preferencia por YOLOv8m, se fundamenta en su capacidad para proporcionar resultados eficientes y precisos en la tarea de detección de objetos, contribuyendo así al éxito general del sistema.

Tabla 18. Versiones de Yolov8. [65]

Modelo	**Size(pixels)**	**mAP50-95**	**Speed CPU ONNX(ms)**	**Params(M)**	**FLOPs(B)**
YOLOv8n	640	37.3	80.4	3.2	8.7
YOLOv8s	640	44.9	128.4	11.2	28.6
YOLOv8m	640	50.2	234.7	25.9	78.9
YOLOv8l	640	52.9	375.2	43.7	165.2

e. Descripción de los modelos de IA utilizados en Yolov8m

- ***Redes Convolucionales Profundas (CNN).*** Las CNN forman la base de YOLOv8m. Son redes neuronales que se especializan en el procesamiento de imágenes y datos espaciales. YOLOv8 utiliza una arquitectura CNN avanzada llamada CSPNet que es eficiente y precisa.

- ***Aprendizaje Profundo.*** YOLOv8m utiliza técnicas de aprendizaje profundo para entrenar sus modelos. Esto implica entrenar la red neuronal con grandes cantidades de datos para que pueda aprender a identificar objetos de forma automática.

- ***Detección de Objetos por Regresión.*** YOLOv8m utiliza un enfoque de regresión para la detección de objetos. En lugar de clasificar cada píxel de la imagen, YOLOv8m predice directamente la ubicación y la clase de los objetos presentes.

- ***Attention Mechanisms.*** YOLOv8m también utiliza mecanismos de atención para enfocarse en las partes más relevantes de la imagen para la detección de objetos. Esto ayuda a mejorar la precisión y la eficiencia del modelo.

- ***Modelo de Backbone.*** YOLOv8m ofrece diferentes modelos de backbone, cómo EfficientNet y MobileNet, que se pueden elegir según la necesidad de equilibrio entre precisión y velocidad.

- ***Modelo de Decodificador.*** El decodificador de YOLOv8m utiliza una arquitectura de "cuello de botella" para procesar las características de la imagen y generar las predicciones finales.

3.1.3 Arquitectura del sistema

A continuación, en la Figura 13 se detalla la arquitectura del sistema:

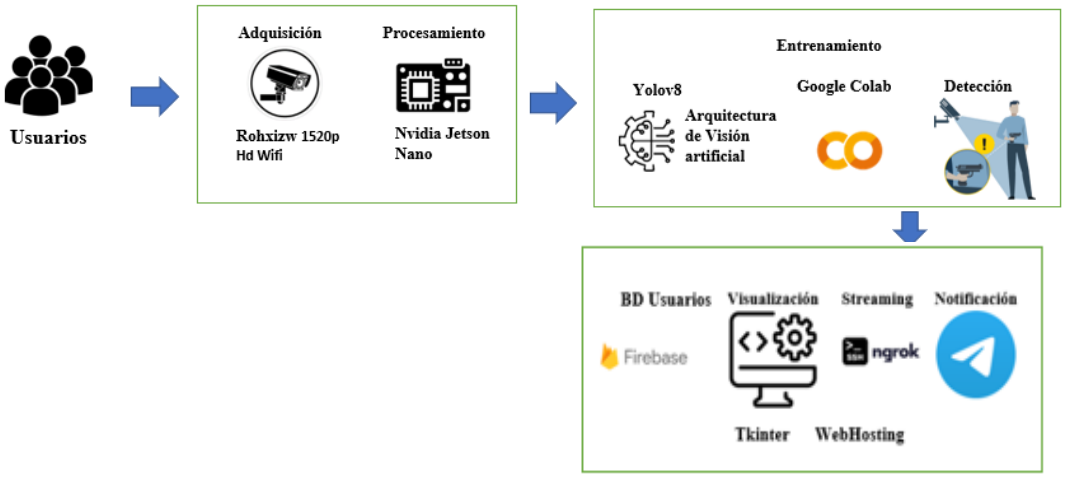


Figura 13. Arquitectura del sistema

a. Etapa 1: Adquisición y procesamiento de datos

- ***Adquisición.*** Para tener acceso al sistema de monitoreo atreves de la cámara de videovigilancia, el usuario debe acceder a una interfaz gráfica con sus credenciales.

- ***Procesamiento:*** Una vez que la cámara captura las imágenes del video en tiempo real, estos datos visuales son transmitidos al microcomputador NVIDIA Jetson Nano. El propósito de esta etapa es realizar una manipulación detallada de la información visual, para la detección. La capacidad de la NVIDIA Jetson Nano para llevar a cabo este análisis en tiempo real garantiza una respuesta dinámica y ágil en la detección de delitos, contribuyendo así a la robustez y eficacia del sistema en su conjunto.

b. Etapa 2.- Entrenamiento

En la etapa de entrenamiento, una vez que se ha preparado el conjunto de datos, se somete a un procedimiento avanzado que implica el uso de técnicas avanzadas de inteligencia artificial, destacando la utilización de redes neuronales. Este modelo de aprendizaje profundo demuestra la capacidad de identificar los parámetros seleccionados para la detección de delitos. Durante este proceso, la red neuronal se ajusta y perfecciona al exponerse a un conjunto diverso y representativo de datos de entrenamiento, lo que permite que el sistema adquiera la habilidad de generalizar y reconocer patrones complejos con una alta precisión, contribuyendo de manera significativa a la efectividad y confiabilidad del sistema.

c. Etapa3.- Almacenamiento, visualización y notificación

- **Almacenamiento.** Las credenciales de los usuarios se han almacenado en Firebase con el propósito de que solo aquellas personas registradas en el sistema tengan la capacidad de visualizar en tiempo real lo que está ocurriendo.

- **Visualización.** Al ingresar con sus credenciales el usuario puede visualizar el video del local comercial en tiempo real. La herramienta Ngrok es utilizada como un puente que permite compartir los servicios locales a internet, de tal manera que el sistema sea remoto.

- **Notificación.** Si el sistema capta a través de la cámara uno de los elementos usados por criminales en situaciones delictivas, se envía una notificación con una screenshot de la actividad sospechosa y un mensaje donde proporciona un detalle del elemento detectado y la ubicación del local comercial.

3.1.4 Descripción del sistema de detección

Se presenta la propuesta del sistema de detecciones de situaciones delictivas en la Figura 14 , el cual está compuesto por diversas etapas. En la Figura 15 se detalla el proceso del sistema implementado por medio de un diagrama de bloques.

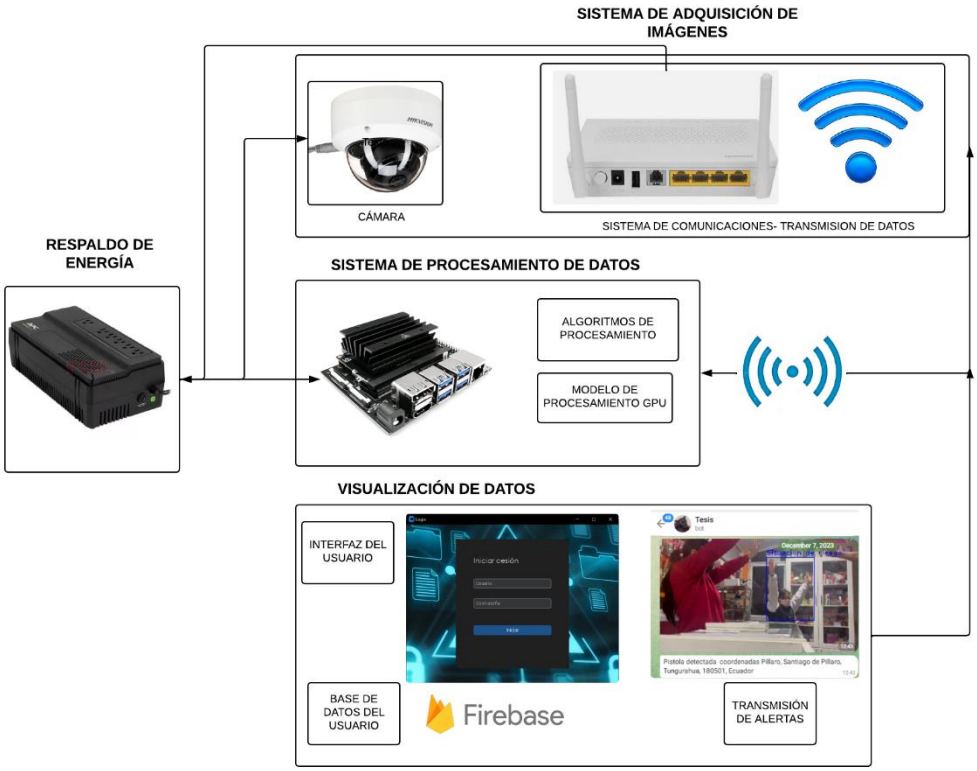


Figura 14. Esquema General del sistema

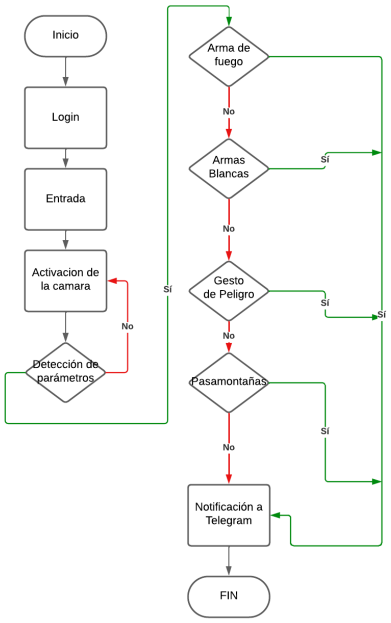


Figura 15. Diagrama de bloques del sistema

3.1.5 Diseño del sistema

La implementación del modelo entrenado se llevó a cabo utilizando el lenguaje de programación Python como se puede ver en el Anexo J en conjunto con el entorno de desarrollo Visual Studio Code. A continuación, se proporciona un desglose detallado de las acciones realizadas en el código para la programación del sistema.

a. Bibliotecas requeridas

En el desarrollo de este sistema, se emplearon diversas bibliotecas fundamentales para potenciar la funcionalidad y eficiencia del software. La Tabla 19 detalla exhaustivamente las librerías seleccionadas, entre las cuales destacan numpy, opencv, ultralytics, sort, itertools, telebot y geopy.

La elección y combinación estratégica de estas librerías se fundamenta en su capacidad colectiva para proporcionar un marco sólido y versátil que potencia las capacidades del sistema en diversos aspectos clave, desde el procesamiento de datos hasta la comunicación con los usuarios y la integración de inteligencia artificial avanzada.

Tabla 19. Bibliotecas requeridas del sistema

Bibliotecas	Características
NumPy	Es una biblioteca de Python utilizada para realizar operaciones numéricas y matriciales de manera eficiente. Se utiliza comúnmente para manipular arreglos y matrices, lo que resulta fundamental en operaciones relacionadas con imágenes y datos numéricos.
OpenCV (Open Source Computer Vision)	Es una biblioteca especializada en visión por computadora y procesamiento de imágenes. Se utiliza para leer, procesar y manipular imágenes, así como para aplicar técnicas avanzadas de visión por computadora.
Ultralytics:	Una biblioteca que proporciona implementaciones eficientes y potentes para algoritmos de visión por computadora. En este caso, se importa el módulo yolo, que es una implementación del algoritmo de detección de objetos yolo (you only look once), utilizado para identificar y localizar objetos en imágenes.
SORT (Simple Online and Realtime Tracking):	Es un algoritmo de seguimiento de objetos en tiempo real. La biblioteca "sort" proporciona una implementación de este algoritmo para el seguimiento de objetos en secuencias de imágenes o vídeos.
Itertools:	Proporciona funciones eficientes para trabajar con iteradores y generadores en python. Se puede utilizar para realizar operaciones combinatorias y manipulaciones avanzadas con iterables.
Telebot:	Es una biblioteca para interactuar con el api de bot de telegram. Permite la creación y gestión de bots de telegram, facilitando la implementación de funciones de chat y automatización de tareas.
Geopy:	Es una biblioteca que proporciona servicios de geocodificación y búsqueda de ubicaciones basados en direcciones o coordenadas geográficas. Nominatim es un servicio de geocodificación de openstreetmap, y está importación permite la obtención de información geográfica a partir de direcciones.

- ***Importación de bibliotecas en la programación***

La importación de las librerías correspondientes se observa en la Tabla 20.

Tabla 20. Importación de librerías

```
import numpy as np
import cv2
from ultralytics import YOLO
from sort import Sort
import itertools
```

b. Adquisición de la imagen

Para que el sistema capte la imagen que será procesada del video en tiempo real se sigue una serie de pasos que se muestran en la Figura 16.

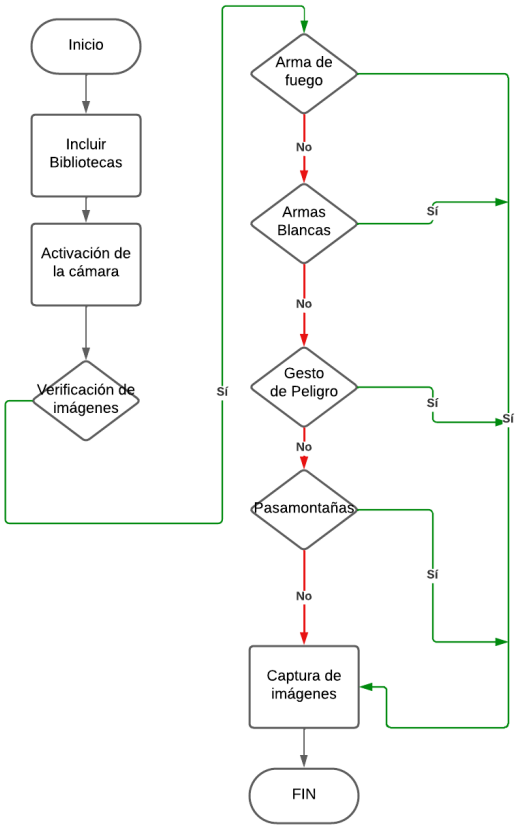


Figura 16. Pasos para la adquisición de imágenes

c. Aplicación de funciones

- ***Función para la sincronización con Telegram***

Para las notificaciones, se eligió utilizar la plataforma de Telegram. En el contexto de esta aplicación, se usó un "bot", y para su configuración, se ha decidido utilizar una herramienta específica conocida como BotFather. Este gestor facilita la creación de bots destinados a diversos propósitos, incluyendo su implementación en grupos y otras aplicaciones específicas. Con BotFather se logra una configuración sencilla y personalizada que se ajusta a las necesidades particulares de la aplicación, es necesario

buscar al bot en Telegram e iniciarlo, una vez iniciado se crea un nuevo bot y automáticamente se obtiene el TOKEN el cual se puede visualizar en Figura 17.

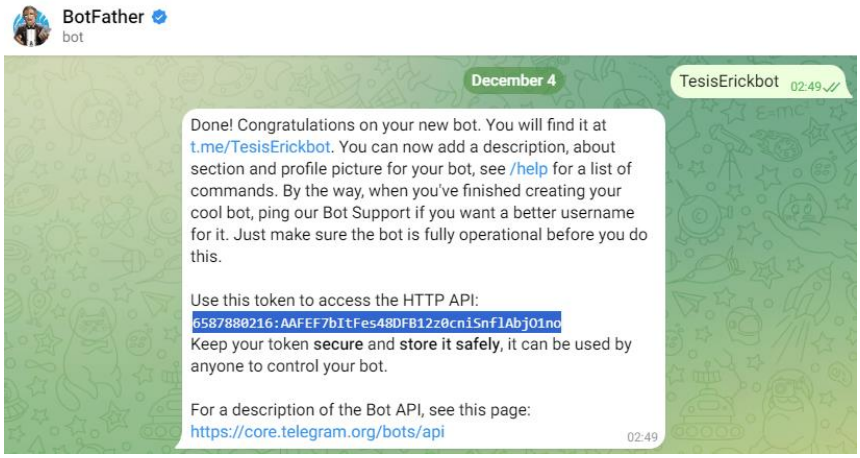


Figura 17. Bot de telegram

En el código que se representa en la Tabla 21 se indica el uso del Token obtenido a través de BotFather para crear una instancia del bot. La creación de esta instancia del bot es esencial para habilitar su funcionamiento y permitir la interacción con usuarios y grupos de Telegram.

Tabla 21. Instanciación del bot

```
TOKEN = '5955691907:AAHr3oFVhPDRM8LZgkln2x2L0r5uXdiUY0c'
bot = telebot.TeleBot(TOKEN)
```

- ***kFunciones de envió de notificaciones***

Si el sistema detecta uno de los parámetros para la cual fue entrenado, envía la notificación al Telegram. Este enfoque asegura que las alertas se generen de manera adecuada, sin sobrecargar el sistema con notificaciones innecesarias y garantizando una respuesta oportuna en el caso de detecciones relevantes. El código usado se lo puede ver en la Tabla 22.

Tabla 22. Funciones de envió de alertas

```
def enviar_alerta(chat_id, mensaje):
 global ultima_ejecucion
    tiempo_actual = time.time()
```

```
    if tiempo_actual - ultima_ejecucion >= 10:
        bot.send_message(chat_id, mensaje)
        ultima_ejecucion = tiempo_actual
        with open("captura.jpg", 'rb') as photo:
            bot.send_photo(chat_id, photo)
def enviar_alerta2(chat_id, mensaje):
    global ultima_ejecucion2
    tiempo_actual = time.time()
    if tiempo_actual - ultima_ejecucion2 >= 10:
        bot.send_message(chat_id, mensaje)
        ultima_ejecucion2 = tiempo_actual
        with open("captura2.jpg", 'rb') as photo:
            bot.send_photo(chat_id, photo)
def enviar_alerta3(chat_id, mensaje):
    global ultima_ejecucion2
    tiempo_actual = time.time()
    if tiempo_actual - ultima_ejecucion2 >= 10:
        bot.send_message(chat_id, mensaje)
        ultima_ejecucion2 = tiempo_actual
        with open("captura3.jpg", 'rb') as photo:
            bot.send_photo(chat_id, photo) }
```

- ***Configuración de la cámara en el código***

Para configurar la cámara se necesita un software denominado YI IOT el cual se escanea y ya se conecta directamente la cámara al internet.

Figura 18. Configuración de la cámara

- ***Captura de Video***

En este proceso comienza la inicialización de un objeto de captura de video denominado "cap", el cual se encarga de abrir la cámara seleccionada para su uso posterior. Este proceso establece la conexión con la cámara, permitiendo la adquisición de secuencias de vídeo desde la misma.

cap = cv2.VideoCapture(0)

- ***Carga del Modelo YOLO***

Se procede a cargar un modelo YOLO desde el archivo denominado "best5.pt". Esta acción implica la incorporación del modelo YOLO, permitiendo su utilización posterior en el proceso de detección de objetos.

model = YOLO("best5.pt")

- ***Inicialización de rastreadores SORT***

Se procede a la inicialización de tres rastreadores SORT. Esta acción implica la preparación y configuración de tres instancias del algoritmo de seguimiento SORT, que desempeña un papel crucial en el seguimiento y la identificación de objetos.

tracker = Sort() tracker2 = Sort() tracker3 = Sort()

- ***Obtención de Fotograma***

Se realiza la lectura de un fotograma proveniente del objeto de captura de video. El estado ("status") indicará si el fotograma se ha leído correctamente. Este procedimiento implica la extracción de un cuadro de imagen del flujo de video, y el estado asociado determina si la operación fue exitosa o no.

status, frame = cap.read()

- ***Detección de Objetos con YOLOv8m***

Se emplea el modelo YOLOv8m para llevar a cabo la detección de objetos en el fotograma actual. La configuración stream=True podría estar vinculada a la posibilidad de realizar transmisión de video en tiempo real. Además, se especifica una lista de clases de interés mediante la variable classes=[0,1,2]. Este enfoque asegura que la detección se realice únicamente para las clases de objetos específicos enumeradas en la lista.

results = model(frame, stream=True, classes=[0,1,2])

- ***Filtrado de Detecciones***

En la Tabla 23 se lleva a cabo un proceso de filtrado de las detecciones, considerando tanto la confianza (conf) como la clase del objeto (cls). Específicamente, se están filtrando objetos pertenecientes a las clases 0, 1 y 2, utilizando umbrales de confianza específicos para cada una de ellas. Este enfoque permite establecer criterios precisos para la selección de detecciones relevantes con base en la confianza y la clase de los objetos detectados.

Tabla 23. Filtrado de detecciones

```
probs = res.probs filtered_indices = np.where((res.boxes.conf.cpu().numpy() >
0.30)&(((res.boxes.cls.cpu().numpy() == 1))))[0]
filtered_indices2 = np.where((res.boxes.conf.cpu().numpy() >
0.30)&(((res.boxes.cls.cpu().numpy() == 0))))[0]
```

- ***Obtención de Cajas delimitadoras (Bounding Boxes)***

En la Tabla 24 se procede a obtener las cajas delimitadoras (Bounding Boxes) correspondientes a las detecciones previamente filtradas. Este paso implica la identificación y extracción de las coordenadas que definen las áreas en las cuales los objetos detectados se encuentran delimitados en la imagen.

Tabla 24. Bounding boxes

```
boxes = res.boxes.xyxy.cpu().numpy()[filtered_indices].astype(int)
boxes2 = res.boxes.xyxy.cpu().numpy()[filtered_indices2].astype(int)
```

- ***Seguimiento con el algoritmo SORT***

En la Tabla 25 se emplea el algoritmo SORT con el fin de llevar a cabo el seguimiento continuo de las cajas delimitadoras a medida que se suceden los fotogramas. Este proceso garantiza la continuidad en el rastreo de las áreas definidas por las cajas, permitiendo un seguimiento preciso y consistente de los objetos detectados a lo largo del flujo de video.

Tabla 25. Seguimiento con el algoritmo SORT

```
tracks = tracker.update(boxes) tracks = tracks.astype(int)
tracks2 = tracker2.update(boxes2) tracks2 = tracks2.astype(int)
```

- ***Dibujar texto y rectángulo en el fotograma***

El código de la Tabla 26 muestra que se incorpora un texto identificativo por ejemplo ("Pistola") sobre el objeto detectado y procede a trazar un rectángulo alrededor de la región de interés en el fotograma. Esta acción visual no solo añade una etiqueta descriptiva al objeto detectado, sino que también resalta de manera gráfica la ubicación exacta de la región identificada en la imagen.

Tabla 26. Dibujar texto y rectángulo

```
cv2.putText(img=frame, text=f"Pistola", org=(xmin, ymin-10),
fontFace=cv2.FONT_HERSHEY_PLAIN, fontScale=2, color=(0,0,255), thickness=2)
cv2.rectangle(img=frame, pt1=(xmin, ymin), pt2=(xmax, ymax), color=(0, 0, 255), thickness=2)
```

- ***Creación de Mensaje y Captura de Imagen***

El siguiente código genera un mensaje que informa sobre la detección del elemento utilizado en el acto delictivo, incluyendo las coordenadas geográficas del local comercial, y guarda el fotograma con la detección como "captura.jpg". Este proceso no solo comunica la detección de manera explícita, sino que también almacena visualmente la evidencia mediante la captura de la imagen identificada para referencias posteriores.

mensaje2 = 'Pistola detectada '+ " coordenadas "+ direccion
cv2.imwrite("captura.jpg", frame)

- ***Envío de alerta***

Este código envía una alerta al destinatario cuyo ID es '1338645187' junto con el mensaje creado. Esta acción facilita la notificación inmediata y específica a la persona

o entidad designada sobre la detección de la pistola, asegurando una respuesta rápida y efectiva.

enviar_alerta ('1691212127', mensaje2)

Para tener el ID del telegram hay que buscar el bot llamado Json Dump Bot e inicializar para que éste otorgó la ID del telegram del usuario que quiere recibir la alerta como se puede ver en la Figura 19.

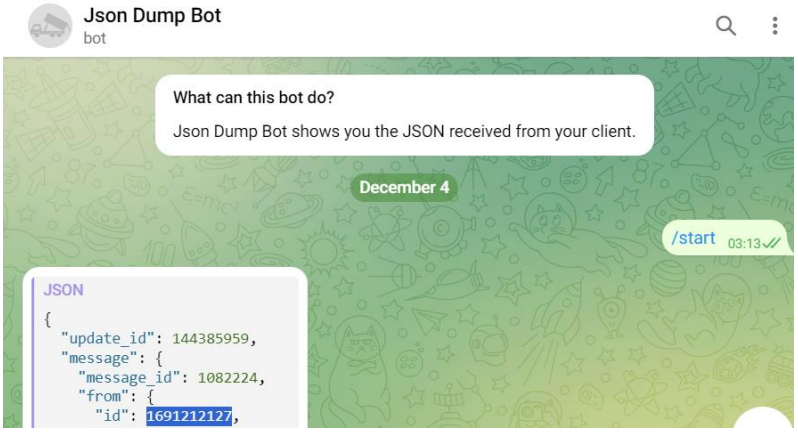


Figura 19. ID de Telegram

d. Activación de CUDA

CUDA (Compute Unified Device Architecture) es una plataforma de computación paralela desarrollada por NVIDIA. Esta tecnología permite utilizar las unidades de procesamiento gráfico (GPU) de NVIDIA para realizar tareas de cómputo general junto con las operaciones gráficas tradicionales. En lugar de limitar las GPU únicamente a gráficos, CUDA permite a los desarrolladores aprovechar el poder de procesamiento masivo de las GPU para realizar cálculos intensivos de manera más eficiente que en las CPU convencionales. Para configurarla primero es necesario llenar el formulario con las especificaciones de la NVIDIA JETSON NANO que se ven en Figura 20. [66]

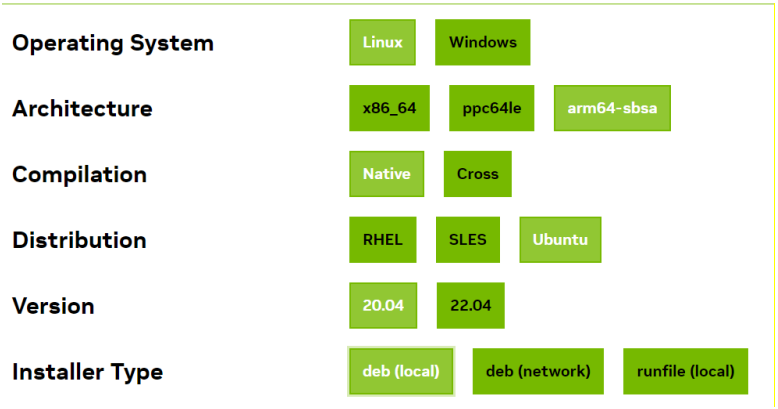


Figura 20. Formulario para descargar CUDA [66]

Una vez llenado el formulario, la página despliega los comandos a seguir para instalar CUDA.

-wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/sbsa/cuda-ubuntu2004.pinsudo
-mv cuda-ubuntu2004.pin /etc/apt/preferences.d/cuda-repository-pin-600
-wget https://developer.download.nvidia.com/compute/cuda/12.3.2/local_installers/cuda-repo-ubuntu2004-12-3-local_12.3.2-545.23.08-1_arm64.debsudo dpkg -i cuda-repo-ubuntu2004-12-3-local_12.3.2-545.23.08-1_arm64.deb
-sudo cp /var/cuda-repo-ubuntu2004-12-3-local/cuda-*-keyring.gpg /usr/share/keyrings/
-sudo apt-get update
-sudo apt-get -y install cuda-toolkit-12-3

e. Instalación de dependencias

Con el siguiente comando se instala la biblioteca de Python ultralytics, que implementa el modelo de detección de objetos YOLOv8. La versión 8.0.124 es una versión específica de la biblioteca con algunas características y correcciones de errores. Los resultados se presentan en la Figura 21:

```
[ ] !pip install ultralytics==8.0.124

    Collecting ultralytics
      Downloading ultralytics-8.0.207-py3-none-any.whl (645 kB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 645.2/645.2 kB 12.0 MB/s eta 0:00:00
    Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (3.7.1)
    Requirement already satisfied: numpy>=1.22.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.23.5)
    Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (4.8.0.76)
    Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (9.4.0)
    Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (6.0.1)
```

Figura 21. Resultados de la instalacion de la biblioteca ultralytics

Para la implementación del modelo de redes neuronales convolucionales basada en la arquitectura de CPSNET se realizaron dos programaciones, una se realizó con 20 épocas y con 45 épocas. YOLOv8m utiliza una combinación de técnicas de aumento

de datos, regularización, dropout, transferencia de aprendizaje y entrenamiento en conjuntos de datos grandes para evitar el sobreajuste y mejorar su generalización a nuevas imágenes.

f. Dataset

En la búsqueda de una base de datos adecuada para el entrenamiento en reconocimiento de delitos, se priorizó la selección de conjuntos de datos que proporcionaran una amplia variedad de imágenes. Este enfoque se adoptó con el propósito de mejorar la capacidad del modelo para generalizar y reconocer armas. La estructura de las bases de datos adoptadas se organiza de manera específica como se observa en la Figura 30 para respaldar las fases críticas del proceso de entrenamiento.

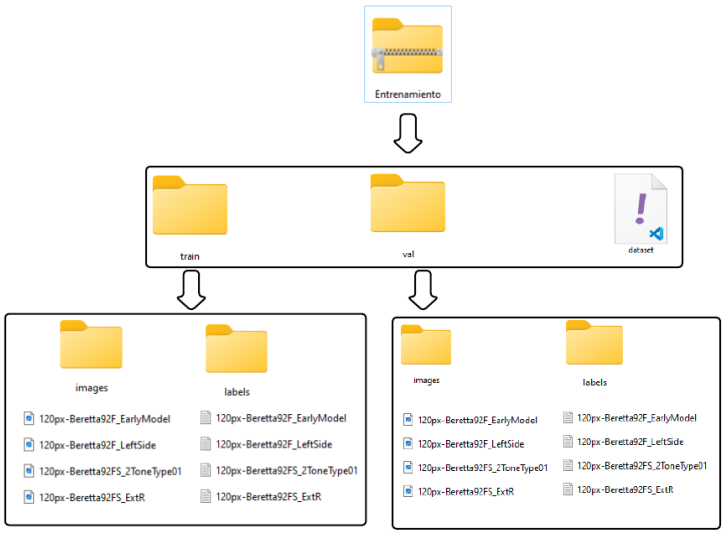


Figura 22. Arquitectura de la base de datos.

La base de datos está constituida por dos componentes principales, ambos dirigidos al proceso de entrenamiento. El primer componente es la carpeta de entrenamiento ("train"), que se divide en dos subcarpetas clave: "images" y "labels". En la subcarpeta "images", se guardan imágenes que representan los parámetros de interés. Por otro lado, la subcarpeta "labels" contiene los parámetros correspondientes a las coordenadas de las bounding boxes, obtenidas mediante la etiquetación de las imágenes en la herramienta Labelme. La segunda componente se refiere a la carpeta de validación ("val"), que también se organiza en las subcarpetas "images" y "labels". En contraste con la carpeta de entrenamiento, las imágenes aquí presentes son

desconocidas para la red neuronal durante el proceso de entrenamiento. Esta diferencia es crucial, ya que estas imágenes se emplean para evaluar el rendimiento del modelo frente a datos que no ha visto previamente. Por ende, esta fase de validación se convierte en un indicador fundamental del nivel de desempeño del modelo, asegurando una evaluación rigurosa de su capacidad para generalizar y reconocer delitos de manera efectiva.

- ***Características de la primera base de datos para el entrenamiento.*** En el proceso de entrenamiento de la red neuronal convolucional, se ha logrado identificar y emplear un conjunto de datos específico con atributos definidos. El conjunto de datos seleccionado es conocido como " Object Detection Binary Classifiers methodology based on deep learning to identify small objects handled similarly". Este conjunto de datos tiene las siguientes características representadas en la Tabla 27.

Tabla 27. Características del dataset

Características	**Detalle**
Numero de imágenes	Se compone de un total de 4710 imágenes destinadas al reconocimiento de armas.
Dimensionalidad	Las imágenes se presentan en una dimensión, lo que implica un formato unidimensional para cada imagen en la base de datos.
Parámetros	La base de datos aborda 5 parámetros, de las cuales se usarán las de armas blancas y armas de fuego

- ***Características de la segunda base de datos.*** Con el propósito de tener el mejor funcionamiento en el lugar de aplicación, se creó una base de datos a partir de simular eventos delictivos en el local. Este conjunto de datos tiene las siguientes características representadas en la Tabla 28.

Tabla 28. Carácterísticas de base de datos 2

Características	Detalle
Numero de imágenes	Se compone de un total de 2000 imágenes destinadas para el reconocimiento de delitos.
Dimensionalidad	Las imágenes se presentan en una dimensión, lo que implica un formato unidimensional para cada imagen en la base de datos.
Parámetros	La base de datos aborda 4 parámetros las cuales son armas de fuego, armas blancas, gestos de peligro y pasamontañas.

g. Etiquetado de imágenes

Se optó por emplear la herramienta Labelme para llevar a cabo el proceso de etiquetado en el cual se cargaron ambos conjuntos de datos. La anotación incluye la identificación de objetos individuales y la delimitación de regiones, siendo esencial para tareas como detección de objetos y segmentación semántica.

Yolo toma las coordenadas espaciales de las cajas delimitadoras. Las coordenadas de la caja delimitadora incluyen la posición del cuadro en la imagen, así como su ancho y alto relativos. Además, cada objeto detectado se asocia con una clase específica, como "pistola", "cuchillo", entre otros.

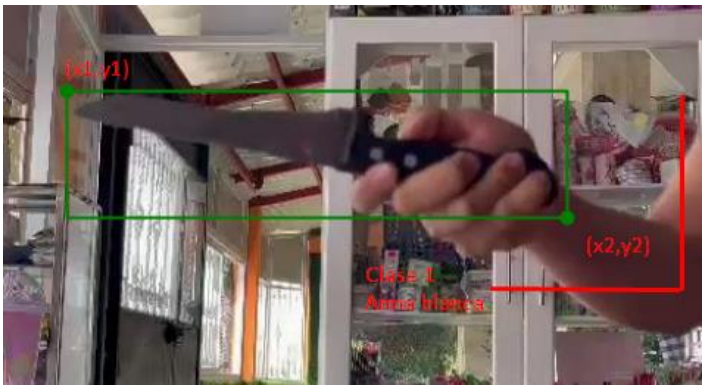


Figura 23. Coordenadas y Clases

La distribución de las clases se puede apreciar de mejor manera en la Tabla 29.

Tabla 29. Distribucion de clases

Clases	**Parámetro**
C0	Cuchillo
C1	Pasamontañas
C2	Pistola
C3	Cuerpo

La estructura de datos que necesita YOLOv8m son la posición en coordenadas rectangulares, posición de base y altura para poder tener un punto intermedio, y las clases. Esta descripción se puede apreciar de mejor manera en la Figura 24.

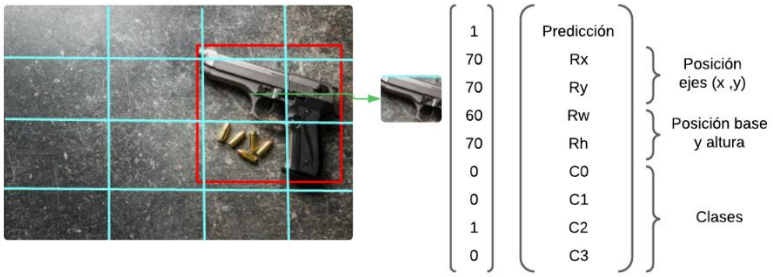


Figura 24. Localizacion del objeto

Se procede a realizar el etiquetado imagen por imagen, este es un proceso manual. Se asigna de manera precisa un rectángulo o polígono delimitador para destacar el objeto de interés como se puede ver en la Figura 25. Estos objetos pueden variar, abarcando desde armas blancas y armas de fuego hasta la identificación de personas que manifiestan encontrarse en una situación de riesgo, indicada mediante una señal visual como levantar los brazos. Este procedimiento no solo implica la tarea de marcar los objetos de interés, sino también la identificación y delimitación precisa de cada uno, contribuyendo así a la generación de datos anotados de alta calidad.

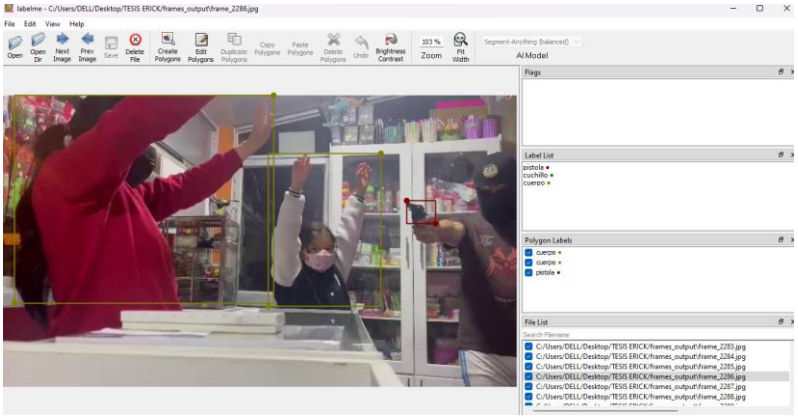


Figura 25. Etiquetado del sistema

El resultado de esto es un conjunto de instrucciones con los parámetros que necesita YOLOv8m para poder ser entrenado, en la Figura 26 se puede un ejemplo la estructura de las instrucciones.


```
{
  "version": "5.3.1",
  "flags": {},
  "shapes": [
    {
      "label": "pistola",
      "points": [
        [
          3.2365145228215813,
          33.34024896265561
        ],
        [
          10.082987551867223,
          30.020746887966816
        ],
        [
          14.85477178423237,
          17.157676348547728
        ],
```


Figura 26. Archivo de salida del proceso de etiquetado

Una vez hecho lo anterior y clasificado en la jerarquía de carpetas como último paso queda cargar la base de datos a Google Colab, como se puede visualizar en la Figura 27.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Figura 27. Proceso de cargar la base de datos

h. Rutas de entrenamiento

Las rutas de los dataset realizados que se utilizará en el modelo de entrenamiento están expresadas a continuación:

- Datos para entrenamiento: *train: /content/drive/MyDrive/Data4/train/*
- Datos para validación: *val: /content/drive/MyDrive/Data4/val/*

i. Arquitectura del modelo interno de YOLOv8m

YOLOv8m utiliza la red neuronal convolucional con la arquitectura CSPNET. Su principal característica es el uso de conexiones de atajo y bloques de red especializados para optimizar el flujo de gradientes y la extracción de características. Esta arquitectura cuenta con bloques CSP los cuales son:

- ***CSP básico:*** El bloque CSP más simple, con una rama de convolución profunda y una rama de convolución ligera.
- ***CSP con corte y mezcla:*** Divide la rama de convolución profunda en dos partes y las mezcla con la rama de convolución ligera.
- ***CSP con atención espacial:*** Incorpora un mecanismo de atención espacial para enfocarse en las áreas relevantes de la imagen.

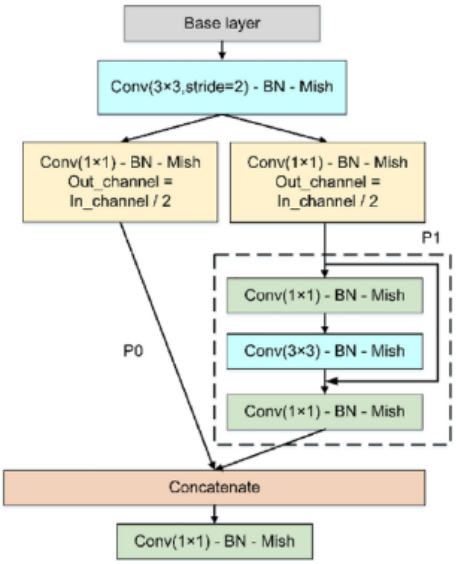


Figura 28. Arquitectura CSPNET

En base a la Figura 28 de la arquitectura, se establecen las primeras capas del modelo que son de convolución y concatenación.

j. Capas del modelo

En la Figura 29 se detallan las capas que tiene la red neuronal convolucional, comenzando desde la capa de entrada, capa oculta y capa de salida. La capa de entrada son los frames o imágenes detectadas por la cámara y de salida es la toma de decisión de la detección.

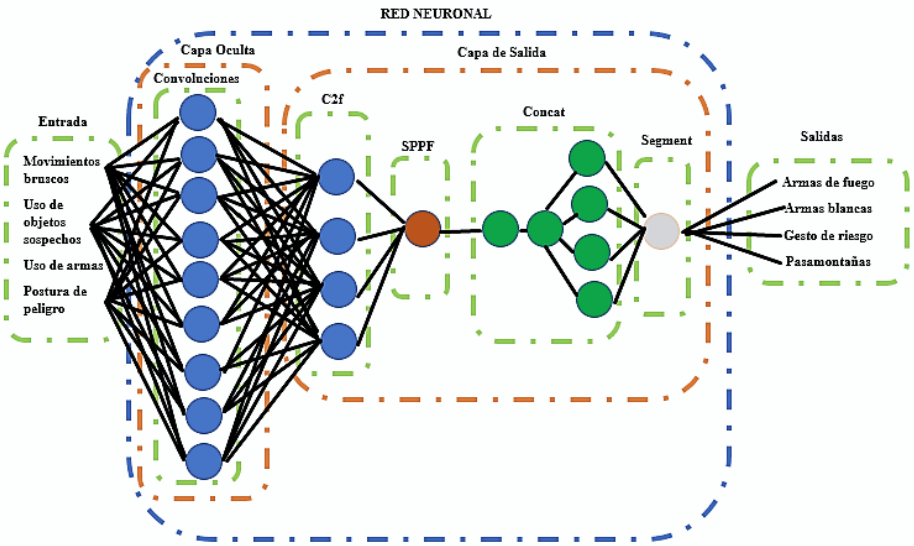


Figura 29. Capas del algoritmo

- ***Capa de Convolución.*** Se implementaron 9 capas de convolución (Conv2D) con el propósito de examinar e identificar patrones característicos o señales de robo en la imagen de entrada. Este proceso se logró mediante la aplicación de filtros que se desplazan a lo largo de cada núcleo o kernel, generando matrices de patrones y condensando la imagen a sus aspectos más significativos. Posteriormente, se aplicó una función de normalización por lotes para estandarizar las activaciones y optimizar el proceso de entrenamiento.

- ***Capa C2f.*** Se implementaron 8 capas C2f, estos bloques incluyen operaciones de convolución y otras transformaciones para aprender representaciones más complejas de las características de entrada.

- ***Capa SPPF.*** Esta capa que realiza agrupamiento espacial piramidal, permitiendo al modelo considerar información contextual a diferentes escalas espaciales en una sola capa.

- ***Capas de Concatenación.*** Se implementaron 6 capas de concatenación con el fin de combinar múltiples conjuntos de características en uno solo, permitiendo al modelo fusionar información de diferentes fuentes.

- ***Capas de Segmentación.*** Esta capa puede estár diseñada específicamente para la tarea de segmentación, que implica asignar una etiqueta a cada píxel de la imagen, indicando a qué clase pertenece.

k. Extracción de características de las imágenes

La robusta arquitectura de YOLOv8m define automáticamente las capas que usa, las cuales se puede observar en la Figura 30, donde se aplican convoluciones, c2f, SPPF, upsample, concat y segment.

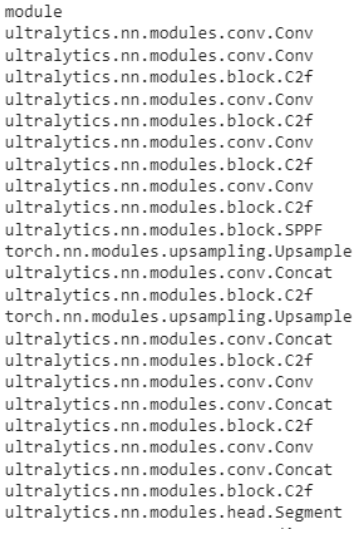


Figura 30. Modelo neuronal

Como resultado en la compilación del modelo neuronal que se puede visualizar en Figura 31, indica que el modelo YOLOv8m consta de 331 capas en total. Estas capas son componentes individuales de la red neuronal, que incluyen capas de convolución, normalización, activación, entre otras. El modelo tiene un total de 27,241,385 parámetros. Los parámetros en un modelo son los pesos y sesgos asociados con las conexiones entre las neuronas. Indica que hay 27,241,369 gradientes asociados con los parámetros.

```
YOLOv8m-seg summary: 331 layers, 27241385 parameters, 27241369 gradients, 110.4 GFLOPs
```

Figura 31. Compilación del modelo de la red neuronal

3.1.6 Entrenamiento del sistema

A continuación, en la Tabla 30, se proporciona un desglose exhaustivo de los parámetros asociados con la red de entrenamiento.

En el proceso de entrenamiento del modelo, se inicia ejecutando el archivo apartado del COLAB que se puede observar en Figura 32. En este procedimiento, se lleva a cabo el entrenamiento con un conjunto de 1800 imágenes, utilizando una proporción de 80% para entrenamiento y 20% para validación como es recomendado. [67]

Se optó por un tamaño de lote de 16 para el proceso por lotes, lo que significa que las imágenes se procesan en grupos de 16. La configuración de las épocas, que determina

el número de repeticiones durante el entrenamiento, inicialmente se configuro con 50 épocas. Sin embargo, gracias a la detección automática de convergencia de YOLOv8m se establece como numero óptimo 45 épocas.

Tabla 30. Parámetros para el entrenamiento

Parámetros	**Características**	**Detalle**
Entrenamiento	1600	Etiquetas de entrenamiento
Etiquetas de validación	400	Validación
IMG	2000	Imágenes
Data	Data/Campo	Conjunto de datos
Batch	16	Determina el tamaño de las muestras que pasaran en la red
Epochs	45	Son épocas totales de entrenamiento
Modelo	Yolov8m	Modelo de entrenamiento

Para llevar a cabo este proceso, se hace uso de un archivo yaml obtenido del etiquetado en LabelMe que almacena las direcciones del conjunto de datos necesario para el entrenamiento. Además, se utiliza una red preentrenada, en este caso, se emplea la arquitectura YOLOv8m.

```
!yolo task=segment mode=train epochs=45 data=/content/drive/MyDrive/Data/dataset.yaml model=yolov8m-seg.pt imgsz=640 batch=16
```

Figura 32. Entrenamiento de la IA

Al iniciar la ejecución del código, se procede a examinar y recuperar las imágenes destinadas tanto al proceso de entrenamiento como al de validación, dando así inicio al ciclo de entrenamiento. La representación visual de este proceso se encuentra detallada en la Figura 33, donde se observa de manera progresiva el avance del entrenamiento a lo largo de las 45 épocas especificadas. Asimismo, en esta representación gráfica se proporciona información detallada sobre el uso de la memoria de la GPU virtual utilizada durante el proceso, ofreciendo una visión integral de la eficiencia y carga de trabajo del sistema durante el entrenamiento del modelo.

```
Downloading https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8m-seg.pt to 'yolov8m-seg.pt'...
100% 52.4M/52.4M [00:00<00:00, 395MB/s]
Ultralytics YOLOv8.0.207 🚀 Python-3.10.12 torch-2.1.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
engine/trainer: task=segment, mode=train, model=yolov8m-seg.pt, data=/content/drive/MyDrive/Data/dataset.yaml, epochs=45, patience=50, batch=2, imgsz=640, sav
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100% 755k/755k [00:00<00:00, 135MB/s]
2023-11-07 05:25:58.028600: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factory: Attempting to register factory
2023-11-07 05:25:58.028659: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory: Attempting to register factory f
2023-11-07 05:25:58.028708: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS factory: Attempting to register factor
Overriding model.yaml nc=80 with nc=3
```

Figura 33. Inicio del entrenamiento

Durante la ejecución de este proceso, se llevan a cabo un total de 45 épocas, cada una de las cuales consume un tiempo promedio de aproximadamente 23 segundos. Durante este intervalo, se observa que el proceso requiere una utilización de memoria GPU de alrededor de 1.72 gigabytes. Es esencial tener en cuenta que la duración y la carga de memoria de la GPU pueden variar significativamente en función del número de parámetros y la cantidad de imágenes ingresadas al sistema. Una representación detallada de este procedimiento se encuentra disponible en la Figura 34, ofreciendo una visión visual del rendimiento del proceso a medida que avanza a lo largo de las épocas especificadas.

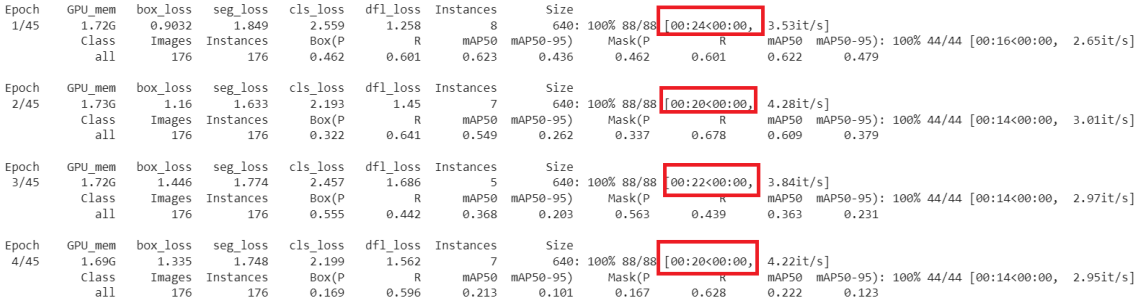


Figura 34. Entrenamiento de la IA por épocas

El proceso de entrenamiento concluye y su resultado se visualiza en la salida de la consola. Específicamente, para el conjunto de datos en cuestión, el servidor de Google ha requerido un tiempo total de 27 minutos para generar el modelo de inferencia. Los archivos resultantes de esta operación son los pesos fundamentales necesarios para la implementación de YOLOv8m en la fase de inferencia, denominados “best.pt” y “last.pt”. A continuación, se detalla la función de cada uno:

Best.pt: Estos son los pesos óptimos recopilados durante el proceso de entrenamiento, representando la configuración que mejor se ajusta a los datos y maximiza el rendimiento del modelo.

Last.pt: Este archivo alberga los pesos correspondientes a la última etapa del entrenamiento. Refleja la configuración del modelo al final del proceso de entrenamiento y puede ser de interés para diversas evaluaciones y comparaciones.

a. Exportación de gráficas

Una vez finalizado el entrenamiento, YOLOV8m proporciona gráficas para el análisis del modelo. Los cuales son representados en la Figura 35.

```
args.yaml                                          MaskF1_curve.png     train_batch3081.jpg
BoxF1_curve.png                                    MaskP_curve.png      train_batch3082.jpg
BoxP_curve.png                                     MaskPR_curve.png     val_batch0_labels.jpg
BoxPR_curve.png                                    MaskR_curve.png      val_batch0_pred.jpg
BoxR_curve.png                                     results.csv          val_batch1_labels.jpg
confusion_matrix_normalized.png                    results.png          val_batch1_pred.jpg
confusion_matrix.png                               train_batch0.jpg     val_batch2_labels.jpg
events.out.tfevents.1699334761.f28362336f07.1518.0 train_batch1.jpg     val_batch2_pred.jpg
labels_correlogram.jpg                             train_batch2.jpg     weights
labels.jpg                                         train_batch3080.jpg
```

Figura 35. Gráficas que proporciona YOLOv8m

- ***Matriz de correlación***

A partir del archivo proporcionado results.csv en YOLOv8m se obtiene una matriz de correlación presente en la Figura 36 entre los conjuntos de datos de entrada, abarcando tanto el total de valores de entrenamiento como la totalidad de los valores asociados a cada una de las variables, la función principal es medir la relación lineal entre cada variable y sus valores pasados.

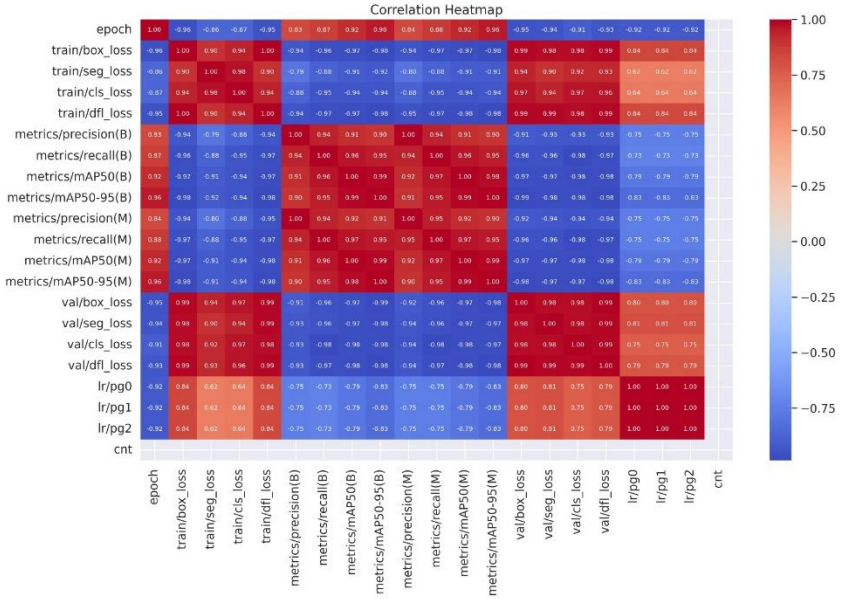


Figura 36 Matriz de correlación

- ***Resultados del entrenamiento.*** Como se observa en la Figura 37 y Figura 38 los resultan con la configuración de 20 y 45 épocas respectivamente, a medida que las épocas aumentan el modelo va generando menos errores:

```
Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
15/20      1.73G      1.022      1.449      1.297      1.306          4        640: 100% 88/88 [00:22<00:00,  3.88it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 44/44 [00:15<00:00,  2.81it/s]
             all        176        176      0.818      0.933      0.952       0.73      0.815      0.928      0.943      0.786

Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
16/20      1.71G     0.9573      1.311      1.113      1.239          8        640: 100% 88/88 [00:22<00:00,  3.90it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 44/44 [00:14<00:00,  3.00it/s]
             all        176        176      0.836      0.919      0.957      0.698      0.833      0.915      0.955       0.77

Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
17/20      1.72G      1.042      1.415      1.236      1.355          6        640: 100% 88/88 [00:20<00:00,  4.37it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 44/44 [00:14<00:00,  3.08it/s]
             all        176        176      0.949      0.903      0.982      0.748      0.944      0.899      0.978      0.772

Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
18/20      1.71G     0.9844      1.512      1.178      1.291          5        640: 100% 88/88 [00:22<00:00,  3.93it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 44/44 [00:14<00:00,  3.04it/s]
             all        176        176      0.687      0.791      0.844      0.653      0.684      0.787       0.84      0.649

Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
19/20      1.71G      0.923      1.317       1.12      1.283          5        640: 100% 88/88 [00:20<00:00,  4.33it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 44/44 [00:14<00:00,  2.96it/s]
             all        176        176      0.815      0.877      0.942      0.748      0.809      0.873      0.937      0.762

Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
20/20      1.73G     0.8779      1.291      1.128      1.262          7        640: 100% 88/88 [00:22<00:00,  3.98it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 44/44 [00:14<00:00,  3.02it/s]
             all        176        176      0.924      0.899      0.975       0.81      0.924      0.897      0.972       0.81
```

Figura 37. Resultados del entrenamiento con 20 épocas.

```
Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
40/45      1.69G     0.5495     0.9058     0.5315      1.083          2        640: 100% 88/88 [00:20<00:00,  4.26it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 44/44 [00:14<00:00,  3.03it/s]
             all        176        176      0.992      0.997      0.995      0.894      0.987      0.992      0.993      0.872

Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
41/45      1.69G     0.5431     0.9752     0.5201      1.058          2        640: 100% 88/88 [00:23<00:00,  3.75it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 44/44 [00:17<00:00,  2.50it/s]
             all        176        176       0.99      0.999      0.995      0.889      0.986      0.995      0.991      0.869

Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
42/45        1.7G      0.522     0.9314     0.5477      1.061          2        640: 100% 88/88 [00:22<00:00,  3.90it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 44/44 [00:15<00:00,  2.90it/s]
             all        176        176      0.943      0.992      0.994      0.894      0.938      0.987       0.99      0.871

Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
43/45      1.69G     0.4877      0.848     0.4919      1.062          2        640: 100% 88/88 [00:20<00:00,  4.21it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 44/44 [00:14<00:00,  2.96it/s]
             all        176        176       0.96      0.993      0.993       0.89      0.955      0.988      0.991      0.872

Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
44/45      1.69G      0.494     0.8671      0.48      1.032          2        640: 100% 88/88 [00:22<00:00,  3.86it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 44/44 [00:15<00:00,  2.87it/s]
             all        176        176      0.934      0.995      0.993      0.897      0.929       0.99      0.991      0.879

Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
45/45      1.69G     0.4856     0.8851     0.4876      1.019          2        640: 100% 88/88 [00:20<00:00,  4.20it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 44/44 [00:14<00:00,  3.01it/s]
             all        176        176       0.94          1      0.994      0.899      0.936      0.995      0.991       0.88
```

Figura 38. Resultados del entrenamiento con 45 épocas.

- ***Precisión de las épocas***

En el análisis de la precisión a lo largo de varias épocas de entrenamiento, se evidencia un patrón ascendente tanto en la precisión del conjunto de entrenamiento como en la de validación. Este comportamiento sugiere un proceso de aprendizaje efectivo en el cual el modelo ha logrado capturar de manera eficaz los patrones inherentes a los conjuntos de datos de entrenamiento y validación. La convergencia simultánea de ambas métricas señala un equilibrio sólido en el proceso de aprendizaje, respaldando la capacidad del modelo para generalizar de manera efectiva a nuevos datos. Sin

embargo, al examinar los resultados representados en la Figura 39, se revela una discrepancia significativa, ya que la precisión en el conjunto de validación se sitúa considerablemente por debajo de la del conjunto de entrenamiento. Esta disparidad plantea preguntas sobre la confiabilidad de los resultados, indicando la necesidad de una evaluación más detallada. Por otro lado, la Figura 40 presenta resultados preliminares prometedores, confirmando la validez y el progreso positivo del modelo durante el entrenamiento. Es importante destacar que, aunque existe la posibilidad de aumentar las épocas de entrenamiento, se observa que los cambios ya no serían abruptos ni representativos en términos de mejora en las métricas.

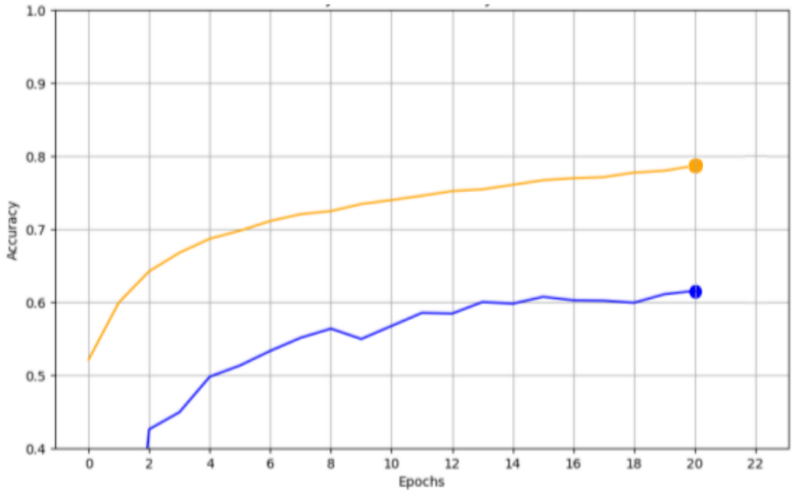


Figura 39. Precisión del modelo con 20 épocas.

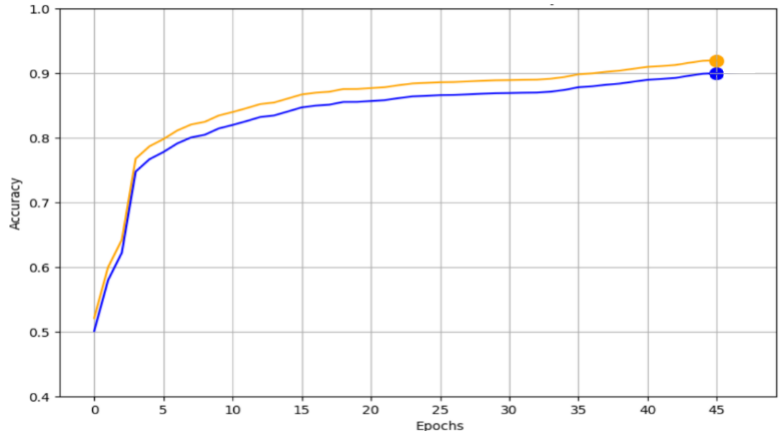


Figura 40. Precisión del modelo con 45 épocas.

- ***Evaluación de los resultados del entrenamiento***

YOLOv8m proporciona de igual manera proporciona una gráfica de la matriz de confusión de los resultados como se puede observar en la Figura 41.

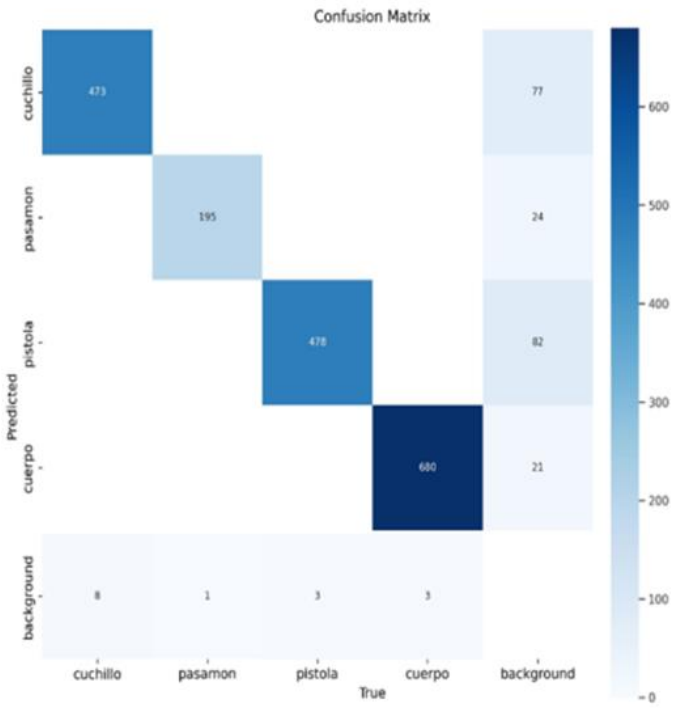


Figura 41. Matriz de correlacion 45 épocas

En la Figura 42 se observan los resultados del proceso de entrenamiento, abordando los parámetros relacionados con la validación del modelo de inteligencia artificial. Estas representaciones muestran claramente que el rendimiento del modelo se atribuye principalmente a la precisión del modelo, la cual experimenta mejoras significativas con el aumento en el número de épocas y es la clave en este sistema de seguridad.

	precision	recall	F1-score	Support
1	0.981	0.995	0.9239	1005
2	0.84	0.999	0.933	960
3	0.90	0.86	0.932	870
4	0.989	0.986	0.901	790
Accuracy			0.9702	1307
macro avg	0.9095	0.9102	0.9017	1307
weighted	0.9357	0.9125	0.9123	1307

Figura 42. Resultados del modelo con 45 épocas

b. Web Hosting

Para la transmisión de datos en tiempo real, se empleó Flask, un marco web ágil y versátil para Python que simplifica el desarrollo de aplicaciones web. Este funciona bajo el patrón de arquitectura de software conocido como "Modelo Vista-Controlador" (MVC). Con este fin se realizó este código que crea una aplicación web utilizando Flask que, cuando se accede a la ruta principal, utiliza Bokeh Server para generar y mostrar visualizaciones interactivas en una plantilla HTML llamada "embed.html". El servidor Flask se ejecuta en el puerto 80 en modo de depuración.

```
from flask import Flask, render_template, Response
app = Flask(__name__)
@app.route('/')
def index():
    with pull_session(url="http://localhost:80/") as session:
        # generate a script to load the customized session
        script = server_session(session_id=session.id, url='http://localhost:80')
        # use the script in the rendered page
    return render_template("embed.html", script=script, template="Flask")
if __name__ == '__main__':
    # runs app in debug mode
    app.run(port=80, debug=True)
```

Con el fin de hacer accesible esta transmisión de datos desde cualquier dispositivo, se implementó un proxy inverso llamado ngrok. Este opera una red global de servidores conocida como ngrok edge, que acepta el tráfico dirigido a los servicios de los clientes en Internet. Las direcciones URL que recibe el tráfico son los puntos de conexión. En contraste con los proxies inversos tradicionales, ngrok no enruta el tráfico hacia los servicios mediante direcciones IP, sino que utiliza un agente. Entonces dentro del programa se ejecuta el comando ngrok http 80 para poder compartir el servicio que este en el puerto 80, como se visualiza en Figura 43. Generando una comunicación con lo diseñado en Flask.

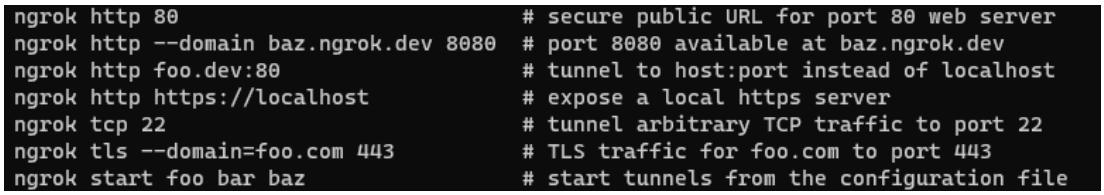


Figura 43. Ngrok Puerto 80

c. Autentificación

Para la autentificación del usuario se usó FireBase. Con el objetivo de que exista un login con usuario y contraseña para ocupar el sistema. Se deben crear los parámetros en la página de FireBase, primero se agrega un documento y luego se agregan los campos, en este caso contraseña y nombre los dos con el tipo String como se puede visualizar en Figura 44.

Figura 44. Documentos y campos de Firebase

En Firebase, no existe un concepto directo de "gráfico entidad-relación", sin embargo, para observar cómo funciona el prototipo se modeló en Figura 45 los parámetros ocupados en la base de datos.

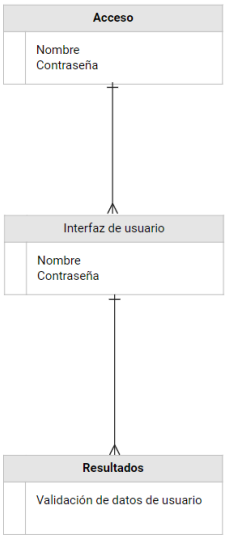


Figura 45. Gráfico Entidad-Relación

d. Interfaz

Se ha desarrollado una interfaz para el microcomputador NVIDIA Jetson Nano, como se ilustra en la Figura 46. Para la seguridad del usuario se ha creado un ingreso con

usuario y contraseña. Una vez que el usuario ha iniciado sesión, se activa la cámara mostrando todo lo que sucede en el local.

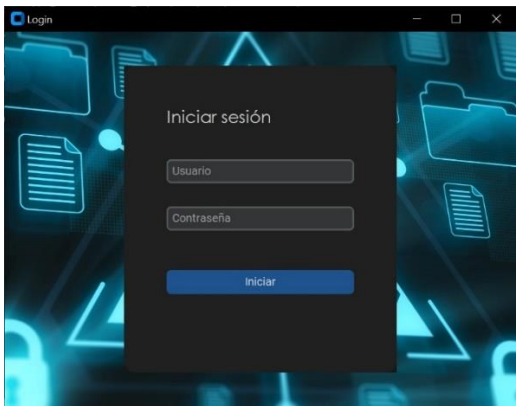


Figura 46. Interfaz Gráfica

La interfaz se enfocó específicamente en la seguridad de la información al crear el sistema de inicio de sesión. Firebase realiza una verificación instantánea para asegurar la concordancia de los datos proporcionados con los registros almacenados. En caso de que la autenticación sea exitosa, el usuario obtiene acceso al sistema; de lo contrario, se le notifica que las credenciales son incorrectas.

3.1.7 Pruebas de Funcionamiento

Para verificar la eficacia del proyecto en la detección de diferentes tipos de objetos, se llevaron a cabo las siguientes pruebas focalizadas en cuatro categorías principales: armas blancas, armas de fuego, pasamontañas y rostros cubiertos con pasamontañas. Dado el amplio espectro de modelos y variaciones dentro de estas categorías, se diseñaron pruebas exhaustivas para evaluar la capacidad del sistema en la detección precisa y oportuna de cada una de ellas.

a. Armas Blancas

Se evaluó la capacidad del sistema para detectar objetos afilados y contundentes que pudieran ser clasificados como armas blancas, por ejemplo, los de la Figura 47. Se utilizaron diversos tipos de cuchillos, navajas y otros objetos punzantes, variando en tamaño, forma y material. Las pruebas se llevaron a cabo en diferentes condiciones de iluminación y ángulos de visualización para simular situaciones reales.

Figura 47. Armas Blancas

b. Armas de Fuego

Se realizaron pruebas con una amplia variedad de armas de fuego, incluyendo pistolas, revólveres y escopetas, las cuales se pueden visualizar en Figura 48, con el objetivo de evaluar la capacidad del sistema para identificar y distinguir estas armas en diversas posiciones y entornos.

Figura 48. Armas de fuego

c. Rostros Cubiertos (Pasamontañas)

Se diseñaron pruebas específicas para evaluar la capacidad del sistema en la detección de rostros humanos cubiertos con pasamontañas u otros elementos de ocultamiento facial, como los que se visualizan en Figura 49. Se utilizaron diferentes tipos de

materiales y colores de pasamontañas, así como variaciones en la iluminación y el ángulo de visión, para evaluar la robustez del sistema ante condiciones adversas.

Figura 49. Pasamontañas

Para asegurar una respuesta rápida y efectiva del sistema, se llevaron a cabo las pruebas en entornos variados y en tiempo real. Esto permitió simular situaciones prácticas y evaluar la capacidad de respuesta del sistema en escenarios dinámicos. Las pruebas se realizaron con la integración del bot de Telegram, lo que facilitó la comunicación instantánea y la activación de la inteligencia artificial para tomar medidas adecuadas ante detecciones relevantes.

- ***Pruebas de funcionamiento en tiempo real y ambientes pregrabados***

Prueba en el escenario 1, como se puede ver en la Figura 50 la prueba de funcionamiento se realizó en un local comercial y aparece un sujeto teniendo un arma blanca en sus manos. Como se puede observar, cuando el sistema detecta el parámetro para el que fue entrenado, manda la notificación a Telegram en el instante que sucedió el presunto delito con la descripción y la ubicación.

Figura 50. Prueba del escenario 1

Prueba en el escenario 2, como se puede ver en Figura 51 la prueba de funcionamiento se realizó en un local comercial y aparece una mujer teniendo un arma blanca en sus manos. Como se puede observar, cuando el sistema detecta el parámetro para el que fue entrenado, manda la notificación a Telegram en el instante que sucedió el presunto delito con la descripción y la ubicación.

Figura 51. Prueba del escenario 2

Prueba en el escenario 3, como se puede ver en Figura 52 la prueba de funcionamiento se realizó en un domicilio y aparece un sujeto teniendo un arma blanca en sus manos. Como se puede observar, cuando el sistema detecta el parámetro para el que fue entrenado, manda la notificación a Telegram en el instante que sucedió el presunto delito con la descripción y la ubicación.

Figura 52. Prueba del escenario 3

Prueba en el escenario 4, como se puede ver en Figura 52 la prueba de funcionamiento se realizó en un local comercial y aparece un sujeto teniendo un arma blanca en sus manos. Como se puede observar, cuando el sistema detecta el parámetro para el que fue entrenado, manda la notificación a Telegram en el instante que sucedió el presunto delito con la descripción y la ubicación.

Figura 53. Prueba del escenario 4

Prueba en el escenario 5, como se puede ver en la Figura 54, la prueba de funcionamiento se realizó en un local comercial y aparece un sujeto teniendo un arma de fuego en sus manos y las personas hacen una señal de que están en peligro. Como se puede observar, cuando el sistema detecta el parámetro para el que fue entrenado, manda la notificación a Telegram en el instante que sucedió el presunto delito con la descripción y la ubicación.

Figura 54. Prueba del escenario 5

Prueba en el escenario 6, como se puede ver en la Figura 55, la prueba de funcionamiento se realizó en un local comercial y aparece un sujeto teniendo un arma de fuego en sus manos y las personas hacen una señal de que están en riesgo. Como se puede observar, cuando el sistema detecta el parámetro para el que fue entrenado, manda la notificación a Telegram en el instante que sucedió el presunto delito con la descripción y la ubicación.

Figura 55. Prueba del escenario 6

Prueba en el escenario 7, como se puede ver en la Figura 56, la prueba de funcionamiento se realizó en un video pregrabado y aparecen múltiples sujetos teniendo un arma de fuego en sus manos y vistiendo pasamontañas. Como se puede observar, cuando el sistema detecta el parámetro para el que fue entrenado, manda la notificación a Telegram en el instante que sucedió el presunto delito con la descripción y la ubicación.

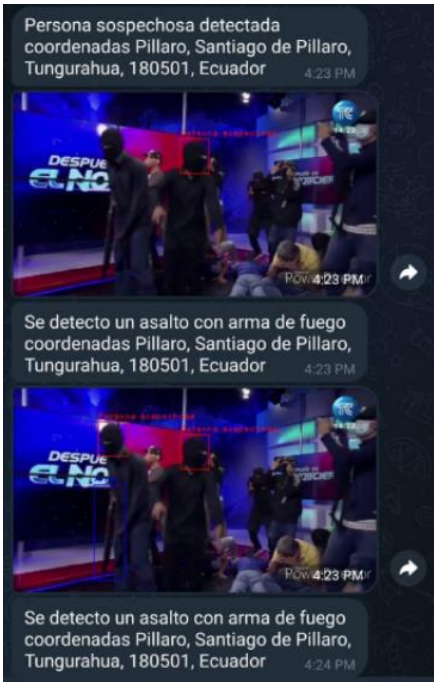


Figura 56. Prueba del escenario 8

Prueba en el escenario 9, como se puede ver en la Figura 57, la prueba de funcionamiento se realizó en un local comercial y aparecen personas haciendo un gesto de que se encuentran en peligro. Como se puede observar, cuando el sistema detecta el parámetro para el que fue entrenado, manda la notificación a Telegram en el instante que sucedió el presunto delito con la descripción y la ubicación.

Figura 57. Prueba del escenario 9

3.2 Validación de resultados

3.2.1 Matriz de confusión de armas blancas

La matriz de confusión de las armas blancas se realizó mediante el análisis de resultados a 100 pruebas, que se resume en la Tabla 31.

Tabla 31. Matriz de confusión

Predicción	Valor positivo	Valor negativo
Predicción positiva	67(VP)	16(FN)
Predicción negativa	3(FP)	14(VN)

• **Interpretación de resultados.** Los datos fueron recopilados a través de pruebas de funcionamiento haciendo una simulación de robo con arma blanca.

Verdaderos positivos: 67

El sistema ha detectado con exactitud a 67 armas blancas.

Verdaderos negativos: 14

El sistema ha realizado una identificación precisa de 14 casos en los cuales aparecen armas blancas.

Falsos positivos: 3

Se registró una instancia en la que el sistema ha categorizado erróneamente a 3 armas blancas cuando, de hecho, no aparecían.

Falsos negativos: 16

En 16 situaciones, el sistema no ha reconocido correctamente armas blancas, indicando que el sistema no detectó una condición que realmente existía.

3.2.2 Matriz de confusión de armas de fuego

La matriz de confusión de las armas de fuego se realizó mediante el análisis de resultados a 100 pruebas, que se resume en la Tabla 32.

Tabla 32. Matriz de confusión

Predicción	**Valor positivo**	**Valor negativo**
Predicción positiva	81(VP)	10(FN)
Predicción negativa	5(FP)	4(VN)

• **Interpretación de resultados.** Los datos fueron recopilados a través de pruebas de funcionamiento haciendo una simulación de robo con armas de fuego.

Verdaderos positivos: 81

El sistema ha detectado con exactitud a 81 armas de fuego.

Verdaderos negativos: 4

El sistema ha realizado una identificación precisa de 4 casos en los cuales aparecen armas de fuego.

Falsos positivos: 5

Se registró una instancia en la que el sistema ha categorizado erróneamente a 5 armas de fuego cuando, de hecho, no aparecían.

Falsos negativos: 10

En 10 situaciones, el sistema no ha reconocido correctamente las armas de fuego, indicando que el sistema no detectó una condición que realmente existía.

3.2.3 Matriz de confusión del gesto de peligro

La matriz de confusión del gesto de peligro se realizó mediante el análisis de resultados a 100 pruebas, que se resume en la Tabla 33.

Tabla 33. Matriz de confusión

Predicción	Valor positivo	Valor negativo
Predicción positiva	60(VP)	17(FN)
Predicción negativa	5(FP)	18(VN)

• **Interpretación de resultados.** Los datos fueron recopilados a través de pruebas de funcionamiento haciendo una simulación donde la víctima hace un gesto de peligro.

Verdaderos positivos: 60

El sistema ha detectado con exactitud a 60 personas haciendo un gesto de peligro.

Verdaderos negativos: 18

El sistema ha realizado una identificación precisa de 18 casos en los cuales aparecen personas haciendo un gesto de peligro.

Falsos positivos: 5

Se registró una instancia en la que el sistema ha categorizado erróneamente 5 gestos de peligro cuando, de hecho, no aparecían.

Falsos negativos: 17

En 17 situaciones, el sistema no ha reconocido correctamente el gesto de peligro, indicando que el sistema no detectó una condición que realmente existía.

3.2.4 Matriz de confusión de personas con pasamontañas

La matriz de confusión de las personas con pasamontañas se realizó mediante el análisis de resultados a 100 pruebas, que se resume en la Tabla 34.

Tabla 34. Matriz de confusión

Predicción	Valor positivo	Valor negativo
Predicción positiva	54(VP)	16(FN)
Predicción negativa	17(FP)	13(VN)

• **Interpretación de resultados.** Los datos fueron recopilados a través de pruebas de funcionamiento haciendo una simulación donde llevan pasamontañas.

Verdaderos positivos: 54

El sistema ha detectado con exactitud a 54 personas con pasamontañas.

Verdaderos negativos: 13

El sistema ha realizado una identificación precisa de 13 casos en los cuales aparecen personas llevando un pasamontaña.

Falsos positivos: 17

Se registró 17 instancias en la que el sistema ha categorizado a personas con pasamontañas, de hecho, no aparecían.

Falsos negativos: 16

En 17 situaciones, el sistema no ha reconocido correctamente a personas con pasamontañas, indicando que el sistema no detectó una condición que realmente existía.

3.2.5 Validación de resultados

a. Cálculos de armas de blancas

- ***Precisión***

$$Presición = \frac{VP}{(VP+FP)} \tag{2}$$

$$Presición = \frac{67}{(67+3)}$$

$$Presición = 0.95$$

- ***Sensibilidad (Recall)***

$$Recall = \frac{VP}{(VP+FN)} \tag{3}$$

$$Recall = \frac{67}{(67 + 16)}$$

$$Recall = 0.81$$

- ***F1_score***

$$F1\_score = 2 * (\frac{Precisión * Recall}{Precisión + Recall}) \quad (4)$$

$$F1\_score = 2 * (\frac{0.95 * 0.81}{0.95 + 0.81})$$

$$F1\_score = 0.87$$

- ***Exactitud***

$$Exactitud = (\frac{VP + VN}{VP + FP + VN + FN} \quad (5)$$

$$Exactitud = (\frac{67 + 14}{67 + 3 + 14 + 16})$$

$$Exactitud = 0.81$$

b. Cálculos de armas de fuego

- ***Precisión***

$$Presición = \frac{VP}{(VP + FP)}$$

$$Presición = \frac{81}{(81 + 5)}$$

$$Presición = 0.94$$

- ***Sensibilidad (Recall)***

$$Recall = \frac{VP}{(VP + FN)}$$

$$Recall = \frac{81}{(81 + 10)}$$

$$Recall = 0.89$$

- ***F1_score***

$$F1\_score = 2 * (\frac{Precisión * Recall}{Precisión + Recall})$$

$$F1\_score = 2 * (\frac{0.94 * 0.89}{0.94 + 0.89})$$

$$F1\_score = 0.91$$

- ***Exactitud***

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

$$Exactitud = (\frac{81 + 4}{81 + 5 + 4 + 10})$$

$$Exactitud = 0.85$$

c. Cálculos de gesto de peligro

- ***Precisión***

$$Presición = \frac{VP}{(VP+FP)}$$

$$Presición = \frac{60}{(60+5)}$$

$$Presición = 0.92$$

- ***Sensibilidad (Recall)***

$$Recall = \frac{VP}{(VP+FN)}$$

$$Recall = \frac{60}{(60+17)}$$

$$Recall = 0.77$$

- ***F1_score***

$$F1\_score = 2 * (\frac{Precisión*Recall}{Precisión+Recall})$$

$$F1\_score = 2 * (\frac{0.92*0.77}{0.92+0.77})$$

$$F1\_score = 0.83$$

- ***Exactitud***

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

$$Exactitud = (\frac{60 + 18}{60 + 5 + 18 + 17})$$

$$Exactitud = 0.78$$

d. Cálculos de pasamontañas

- ***Precisión***

$$Presición = \frac{VP}{(VP+FP)}$$

$$Presición = \frac{54}{(54+17)}$$

$$Presición = 0.76$$

- ***Sensibilidad (Recall)***

$$Recall = \frac{VP}{(VP+FN)}$$

$$Recall = \frac{54}{(54 + 16)}$$

$$Recall = 0.77$$

- ***F1_score***

$$F1\_score = 2 * (\frac{Precisión * Recall}{Precisión + Recall})$$

$$F1\_score = 2 * (\frac{0.76*0.77}{0.76+0.77})$$

$$F1\_score = 0.76$$

- ***Exactitud***

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

$$Exactitud = (\frac{54 + 13}{54 + 17 + 13 + 16})$$

$$Exactitud = 0.67$$

A continuación, se presenta la Tabla 35. Matriz de confusión, que proporciona los valores específicos de los cuatro parámetros establecidos para validar el modelo de

detección de delito. Cada fila de la tabla representa un parámetro distinto, mientras que las columnas corresponden a los datos definidos para medir la eficacia y precisión del modelo en la detección de cada parámetro. La información presentada en esta tabla es fundamental para comprender el desempeño del modelo en diferentes contextos, contribuyendo así a una evaluación completa y detallada de su capacidad predictiva.

Tabla 35. Matriz de confusión

Parámetros	**Precisión**	**Sensibilidad**	**F1-Score**	**Exactitud**
Armas blancas	0.95	0.81	0.87	0.81
Armas de fuego	0.94	0.89	0.91	0.85
Gesto de peligro	0.92	0.77	0.83	0.78
Pasamontañas	0.76	0.77	0.76	0.67
Total	0.89	0.83	0.84	0.78

Con base en los parámetros de validación analizados, se puede concluir que el sistema de detección cuenta con una elevada precisión y sensibilidad, estableciendo su capacidad para determinar especialmente armas, el parámetro F1-score, indica un balance integral que combina la precisión y la sensibilidad. La precisión del sistema es aceptable, sin embargo, el parámetro de pasamontañas es el que con menor valor cuenta, por el contrario, las armas de fuego y armas blancas son más altos debido a que existía un dataset más grande para estas variables. El parámetro de F1-score, ayuda a evaluar de manera porcentual el rendimiento y versatilidad del modelo aplicado en pruebas de entornos reales, este valor se normaliza de 1 a 0 siendo el valor más alto un rendimiento perfecto y el valor más bajo un rendimiento nulo, después de un proceso de calcular el promedio total del sistema del parámetro F1-score dio como resultado un valor de 0.84.

$$Rendimiento\ del\ sistema = F1\_score * 100 \quad (6)$$
$$Rendimiento\ del\ sistema = 0.84 * 100$$
$$Rendimiento\ del\ sistema = 84\%$$

Este porcentaje determina el rendimiento del sistema detector de actos delictivos mediante inteligencia artificial, es fundamental comprender que este parámetro será

muy variado debido a diversos factores como los entornos reales de prueba y los datos característicos adquiridos.

3.3 Presupuesto

El presupuesto del sistema de detección de situaciones delictivas en establecimientos comerciales usando inteligencia artificial se realiza mediante cálculos del presupuesto del diseño y de construcción. Para el diseño se tomó en cuenta las horas empleadas en su fabricación y el sueldo de un Ingeniero en Telecomunicaciones que parte de 858 dólares de acuerdo con el Ministerio de Trabajo de Ecuador, para esto se considera 21 días laborales para conseguir el salario por cada día.

$$\boldsymbol{Salario_{\text{xdía}} = \frac{Salario_{xmes}}{Días\ laborables}} \tag{7}$$

$$Salario_{xdía} = \frac{858}{21}$$

$$Salario_{xdía} = 40.86\$$$

Las horas laborables de acuerdo con el Ministerio de trabajo de Ecuador son 8 y se aplica la siguiente ecuación para obtener el valor por hora de trabajo en dólares.

$$\boldsymbol{Salario_{xhora} = \frac{Salario_{xdia}}{Horas\ laborables}} \tag{8}$$

$$Salario_{xhora} = \frac{40.86}{8}$$

$$Salario_{xhora} = 5.11\$$$

Se ocupó varios materiales para el desarrollo del proyecto los cuales se describen en la Tabla 36.

Tabla 36. Presupuesto

Ítem	Detalle	Valor total
1	NVIDIA JETSON NANO	$200
2	Rohxizw 1520p Hd Wifi	$20
3	UPS 450 W	$84
9	Total	$304

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- La incapacidad de ofrecer notificaciones en tiempo real representa una brecha en la eficacia de estos sistemas, ya que la respuesta inmediata a eventos delictivos es esencial para mitigar pérdidas y garantizar la seguridad de las personas y los activos. Esta constatación subraya la necesidad de explorar alternativas que no solo ofrezcan retrospectiva, sino que también aborden de manera proactiva la seguridad en tiempo real. La integración de tecnologías avanzadas podría ser clave para superar estas limitaciones y proporcionar soluciones más integrales y eficientes en el ámbito de la seguridad en locales comerciales.

- El sistema empleado para la identificación de actividades delictivas se basa en el avanzado algoritmo YOLOv8, que cuenta con un Dataset compuesto por tres categorías principales: armas blancas, armas de fuego, pasamontañas y gestos que indican peligro. Este algoritmo ha demostrado una notable precisión durante las fases de entrenamiento y testeo, alcanzando un resultado de 84% de aceptabilidad en comparación con otros algoritmos de visión artificial disponibles.

- Después de un análisis de las características técnicas como los aspectos económicos, se seleccionó la NVIDIA Jetson Nano como el microordenador por su capacidad de procesamiento. Para la visión, se optó por integrar una cámara Rohxizw 1520p Hd Wifi, elegida por su impresionante resolución y una atractiva relación costo-eficacia. Se incorporó un UPS de 450W para garantizar la continuidad operativa y la integridad de los datos en casos de cortes de energía, con una capacidad de suministro de 7.5 horas. En el entrenamiento del sistema, se utilizó el algoritmo YOLOv8m mediante la plataforma Google Colab, seleccionada por su destacada velocidad de entrenamiento, optimizando así el tiempo en esta fase crucial del desarrollo.

- Luego de las 45 épocas de entrenamiento intensivo, demostrando la eficacia del aprendizaje del modelo. En las pruebas operativas, se evaluó la capacidad del modelo para detectar y clasificar elementos críticos, como armas blancas, armas de fuego, pasamontañas y gestos de peligro. Los resultados fueron alentadores, con una aceptabilidad general del 84%, respaldando la eficacia y fiabilidad del sistema en la detección y clasificación de amenazas potenciales.

4.2 Recomendaciones

- La expansión del dataset utilizado en el proceso de entrenamiento se revela como un factor crítico para obtener resultados superiores en la detección de objetos. A medida que se incrementa el tamaño del dataset, se brinda al modelo una mayor diversidad y riqueza de ejemplos, permitiéndole aprender de manera más completa y generalizada.

- Es importante tener en cuenta que la efectividad de la detección será mayor si se utiliza la misma cámara con la cual se entrenó el modelo. Esto se debe a que cualquier cambio en la resolución o los megapíxeles puede llevar a que el sistema funcione de manera menos precisa.

- La elección de un sistema que integre tensores con Tensor Processing Units (TPU) en lugar de Graphics Processing Units (GPU) se presenta como una estrategia para lograr un procesamiento de visión artificial más rápido y eficaz. La capacidad especializada de las TPUs en tareas específicas, como el procesamiento de datos tensoriales utilizado en la visión por computadora, puede acelerar significativamente el tiempo de procesamiento y mejorar la eficiencia del modelo.

- Un aspecto crucial en la obtención de un conjunto de datos de campo para el entrenamiento es garantizar una iluminación adecuada. Asegurar una iluminación óptima al recopilar muestras del dataset contribuye a la creación de imágenes de alta calidad, facilitando así el proceso de entrenamiento. Imágenes bien iluminadas permiten al modelo aprender patrones de manera

más precisa y generar representaciones más robustas, lo que se traduce en un rendimiento mejorado durante las fases de detección y clasificación.

REFERENCIAS BIBLIOGRÁFICAS

[1] E. Harrington, Navigating System Integration, Canterbury: Security Today, 2023.

[2] N. Tufiño, Tecnologia al servicio de la seguridad, Quito, 2022.

[3] T. Valle, Inseguridad y delincuencia en Ambato, Ambato: Obest, 2022.

[4] E. Moposita, Sistema de seguridad integral para la monitorización de alaras y alerta, Ambato: Carrera de Ingeniería en Electronica y comunicaciones, 2022.

[5] W. Tapia Bastidas, IMPLEMENTACIÓN DE UN PROTOTIPO DE ALARMA COMUNITARIA, Quito: ESCUELA POLITÉCNICA , 2022.

[6] C. Montesdeoca Garcia, « Desarrollo de aplicación móvil de geolocalización para alarma notificacio de paico, Quito, 2022.

[7] K. Chicaiza, Sistema de alarma comunitaria para el mercado San Juan de la, Ambato, 2020.

[8] F. AGUILAR VILLALBA, DESARROLLO DE UN PROTOTIPO DE ALARMA MULTIMODAL COMUNITARIA, Chimborazo , 2018.

[9] S. Russell, Artificial Intelligence: A Modern Approach, 2009.

[10] I. Goodfellow, Deep Learning, 2016.

[11] K. Fajardo Suarez, Estudio de los procesadores para el desarollo de aplicaicones en tiempo real, Cartagena de Indias, 2006.

[12] S. Smith, The scientish and Engineers guide to digital signal processing.

[13] P. Perez, ««Fundamentos básicos del procesamiento de imágenes,» [En línea]. Available: https://www.famaf.unc.edu.ar/~pperez1/manuales/cim/cap2.html.

[14] A. Martínez, ««Bases Teóricas, Digitalización y Análisis de Imágenes,» 2019. [En línea]. Available: https://web.archive.org/web/20100619121419/http://www.seap.es/telepatologia/telepat. [Último acceso: 2024].

[15] B. Escalante, «Procesamiento Digital de Imágenes,» [En línea]. Available: http://lapi.fi-p.unam.mx/wp-content/uploads/PDI_Cap1_Introduccion.pdf.

[16] O. Reyes Ortiz, Técnicas de Inteligencia artificial utilizadas en el procesamiento de imagenes y su aplicacion en el analisis de pavimentos, 2019.

[17] K. Jain, Fundamentals of Digital Image Processing, California, 2018.

[18] D. Forsyth, Computer Vision: A Modern Approach, Champaign, 2015.

[19] C. Bishop, Pattern Recognition and Machine Learning, 2006.

[20] NVIDIA, Módulo y kit de desarrollo Jetson Nano, 2023.

[21] R. P. Foundation, Raspberry Pi, Raspberry Pi, 2020.

[22] C. AI, CORAL DEV BOARD, 2020.

[23] P. Mell, The NIST Definition of Cloud Computing., ational Institute of Standards and Technology (NIST) Special Publication , 2019.

[24] M. A. Notebooks, Azure Notebooks, 2018.

[25] K. Kernels, «Kaggle Official Page,» 2019. [En línea]. Available: https://www.kaggle.com/kernels.

[26] G. Colab, «Welcome to Colab,» 2023. [En línea]. Available: https://colab.research.google.com/notebooks/welcome.ipynb.

[27] H. Thomas, Introduction to Algorithms, The MIT Press, 2018.

[28] J. Redmon, You Only Look Once: Unified, Real-Time Object Detection, IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016., 2016.

[29] W. Liu, SSD: Single Shot Multibox Detector, European Conference on Computer Vision (ECCV) 2016, 2016.

[30] R. Shaoqing , Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Advances in Neural Information Processing Systems (NeurIPS) 2015., 2015.

[31] B. C. Russell, Interviewee, *LabelMe: A Database and Web-Based Tool for Image Annotation.* [Entrevista]. 2020.

[32] KEYLABS, «DATA ANNOTATION PLATFORM,» [En línea]. Available: https://keylabs.ai/. [Último acceso: 2024].

[33] CVAT, «Open Data Annotation Platform,» [En línea]. Available: https://www.cvat.ai/. [Último acceso: 2024].

[34] E. Appel, Security and Loss Prevention: An Introduction, Lexington, 2018.

[35] M. Stamp, I. S. P. a. Practice, Wiley, 2017.

[36] J. Martínez, Integrated Security Systems in Commercial Spaces: A Comprehensive Analysis, 2021.

[37] W. Stallings, Redes y Comunicaciones de Datos: Un Enfoque Práctico, Pearson Educación, 2019.

[38] J. A. Smith, «Factors Influencing the Efficacy of Video Surveillance Systems: A Comprehensive Analysis,» Journal of Security and Surveillance Technology, 2021. [En línea].

[39] E. Johnson, «Comparative Analysis of Three Home Security Systems: A Decision-Making Guide,» *Journal of Security Technology and Applications*, 2022.

[40] M. González, «Security Challenges in Ecuadorian Businesses: A Comprehensive Analysis,» *Journal of Business Security Studies*, 2023.

[41] El Universo, «El universo (Portal Web),» 2023 dicimbre 17 . [En línea]. Available: https://www.eluniverso.com/noticias/seguridad/estos-30-cantones-del-ecuador-concentran-mas-del-70-de-los-crimenes-en-2023-conozca-cuales-son-y-porque-tienen-

altos-indices-de-criminalidad-nota/#:~:text=En%20estas%20muertes%20violentas%20se,5%20de%20diciemb. [Último acceso: 18 diciembre 2023].

[42] A. EFE, «Primicias Ec,» 30 12 2023. [En línea]. Available: https://www.primicias.ec/noticias/seguridad/ecuador-pais-mas-violento-america-latina/. [Último acceso: 09 02 2024].

[43] J. L. G. Cabañas, Manual de Criminología, 7ª edición, Tirant lo Blanch, 2022.

[44] R. P. 4, «Raspberry Pi 4,» [En línea]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.

[45] N. J. Nano™, 2024. [En línea]. Available: https://www.nvidia.com/es-es/autonomous-machines/embedded-systems/jetson-nano/.

[46] C. AI, «Dev Board,» [En línea]. Available: https://coral.ai/products/dev-board.

[47] R. Pi, «Raspberry Pi Camera Module 2,» 2024. [En línea]. Available: https://www.raspberrypi.com/products/camera-module-v2/. [Último acceso: 2024].

[48] R. N. 8, «Redmi Note 8 2021,» [En línea]. Available: https://www.mi.com/es/product/redmi-note-8-2021.

[49] Y. I. YAxa, «Rohxizw cámara de seguridad inalambrica,» [En línea]. Available: https://store.yaxa.co/products/rohxizw-2-camaras-de-seguridad-inalambricas-de-3-mp-para-exteriores-2-4-ghz-y-5-ghz-wifi-camaras-inteligentes-para-interiores-para-seguridad-del/.

[50] F. P. Technologies, «UPS interactiva 500VA/250W, 6 slds, RJ11, torre comp-120V,» [En línea]. Available: https://www.forzaups.com/es/productos/interna/NT-511-esp/.

[51] APC, «UPS BV de APC, 800 VA, AVR, 120 V,» [En línea]. Available: https://www.apc.com/co/es/product/BV800/unidad-easy-ups-bv-de-apc-800-va-avr-120-v/.

[52] APC, «UPS BV de APC, 650 VA, AVR, 120 V,» [En línea]. Available: https://www.apc.com/co/es/product/BV650/unidad-easy-ups-bv-de-apc-650-va-avr-120-v/.

[53] A. Dix y J. Finlay, Interacción humano-computadora, Costa Rica, 2011.

[54] N. Donald, The Design of Everyday Things, 2008.

[55] R. Gonzales, Digital Image Processing, Miami: Pearson, 2018.

[56] S. Attaway, MATLAB: A Practical Introduction to Programming and Problem Solving, 2016.

[57] E. Matthes, Python Crash Course: A Hands-On, Project-Based Introduction to Programming, 2015.

[58] C. Barski, Land of Lisp: Learn to Program in Lisp, One Game at a Time!, 2010.

[59] Microsoft, «Documentación Oficial de Microsoft Azure Notebooks,» Microsoft, [En línea]. Available: https://visualstudio.microsoft.com/es/vs/features/notebooks-at-microsoft/. [Último acceso: 2024].

[60] A. L. Cervantes, Kaggle for Beginners: A step-by-step guide, 2020.

[61] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2019.

[62] J. Redmon, You Only Look Once: Unified, Real-Time Object Detection, 2016.

[63] S. Ren, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, 2015.

[64] W. Liu, SSD: Single Shot MultiBox Detector, 2016.

[65] J. Redmon, You Only Look Once: Unified, Real-Time Object Detection, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[66] NVIDIA, «CUDA,» [En línea]. Available: https://developer.nvidia.com/cuda-downloads.

[67] J. Ortega, Big data, machine learning y data science in Python., Bogotá, 2023.

[68] I. c. office, «INTERNATIONAL STANDARD ISO 9241-210,» First edition, 2010.

[69] I. c. office, «INTERNATIONALSTANDARD ISO/IEC 27400,» First edition , 2022.

[70] A. Pérez, «ISO 22320:2015 - Guidelines for Incident Response in Emergency Management: A Comprehensive Overview,» *International Journal of Emergency Management*, 2016.

[71] C. Mella, «El País,» 10 julio 2023. [En línea]. Available: https://elpais.com/internacional/2023-07-10/la-inseguridad-en-ecuador-escala-a-niveles-historicos-y-se-impone-como-prioridad-del-proximo-gobierno.html. [Último acceso: 18 diciembre 2023].

[72] E. Freire, Ministerio del interior, 29 05 2023. [En línea]. Available: ttps://www.primicias.ec/noticias/en-exclusiva/aumento-robos-violencia-quito-delincuencia-comun.

[73] A. Molina, «ELUNIVERSO,» 30 06 2023. [En línea]. Available: https://www.eluniverso.com/noticias/seguridad/ecuador-entre-los-tres-paises-con-mas-robos-y-asaltos-durante-los-primeros-cuatro-meses-del-2022-segun-encuesta-de-cid-gallup-nota/..

[74] J. Ortega, Big data, machine learning y data science in Pyyhon, 2023.

[75] R. Pi, «Raspberry Pi Camera Module 2,» Raspberry Pi , 2022. [En línea]. Available: https://www.raspberrypi.com/products/camera-module-v2/. [Último acceso: 2023].

[76] Arducam, «Arducam 8 MP Sony IMX219 camera module with M12 lens LS40136 for Raspberry Pi,» Arducam , 2021. [En línea]. Available: https://www.arducam.com/product/arducam-raspberry-pi-camera-v2-8mp-ixm219-b0103/. [Último acceso: 2022].

[77] Logitech, «C270 HD WEBCAM,» Logitech, [En línea]. Available: https://www.logitech.com/es-roam/products/webcams/c270-hd-webcam.960-000947.html.

[78] T. Flores, «primicias.ec,» 23 03 2023. [En línea]. Available: https://www.primicias.ec/noticias/firmas/violencia-ecuador-crimen-asaltos-muertes/.

[79] S. Sánchez, Cloud Computing: Fundamentos y despliegue de un servicio en la nube, Madrid: Universidad Autonoma de Madrid, 2022.

[80] D. Poole, Artificial Intelligence: Foundations of Computational Agents, 2010.

[81] «La hora,» 8 marzo 2023. [En línea]. Available: https://www.lahora.com.ec/esmeraldas/ecuador-con-record-de-inseguridad-en-el-2023/. [Último acceso: 18 diciembre 2023].

[82] HIKVISION, HIKVISION DS-2CE57, HIKVISION, 2020.

[83] E. M. Rojas, Machine Learning: análisis de lenguajes de programación, Revista Ibérica de Sistemas e Tecnologias de Informacion.

[84] E. R. Silva, Entrenamiento de la Red Neuronal Convolucional YOLO para objetos, México, 2020.

ANEXOS

Anexo A. Categorización de variables

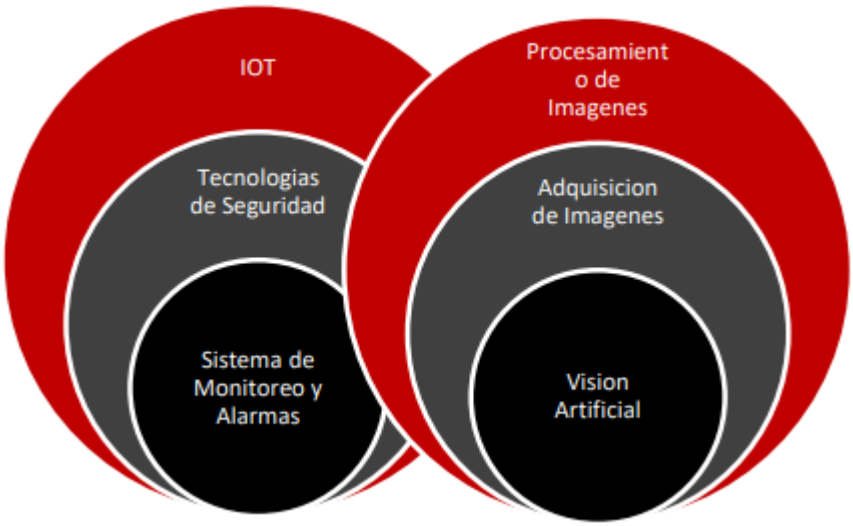
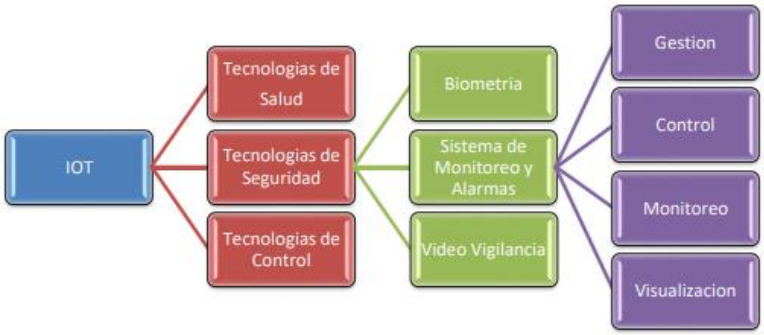


Figura 58. Categorización de variables

Anexo B. Constelación de variables

Constelación de Ideas Variable 1



Constelación de Ideas Variable 2

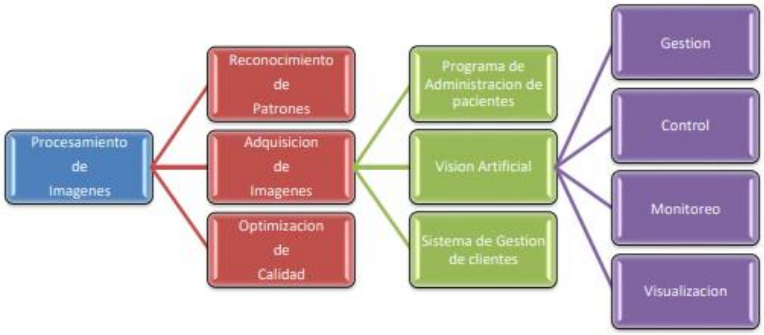


Figura 59. Constelación de variables

Anexo C. NVIDIA JETSON NANO DATASHEET

Jetson Nano module

- 128-core NVIDIA Maxwell GPU
- Quad-core ARM® A57 CPU
- 4 GB 64-bit LPDDR4
- 16 GB eMMC 5.1
- 10/100/1000BASE-T Ethernet

Power

- Voltage Input: 5 V
- Module Power: 5 W–10 W

Environment

- Operating Temperature: -25 C to 80 C*
- Storage Temperature: -25 C to 80 C
- Humidity: 85% RH, 85°C [non-operational]
- Vibration: Sinusoidal 5 G RMS 10 to 500 Hz, random 2.88 G RMS, 5 to 500 Hz [non-operational]
- Shock: 140 G, half sine 2 ms duration [non-operational]

NVIDIA JETSON NANO MODULE
TECHNICAL SPECIFICATIONS

GPU	**128-core Maxwell**
CPU	**Quad-core ARM A57 @ 1.43 GHz**
Memory	**4 GB 64-bit LPDDR4 25.6 GB/s**
Storage	**16 GB eMMC 5.1**
Video Encode	**4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)**
Video Decode	**4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)**
CSI	**12 (3x4 or 4x2) lanes MIPI CSI-2 D-PHY 1.1**
Connectivity	**Gigabit Ethernet**
Display	**HDMI 2.0, eDP 1.4, DP 1.2 (two simultaneously)**
PCIe	**1x1/2/4 PCIe Gen2**
USB	**1x USB 3.0, 3x USB 2.0**
Others	**I²C, I²S, SPI, UART, SD/SDIO, GPIO**
Mechanical	**69.6 mm x 45 mm 260-pin SODIMM Connector**

Figura 60. NVIDIA JETSON NANO DATASHEET [45]

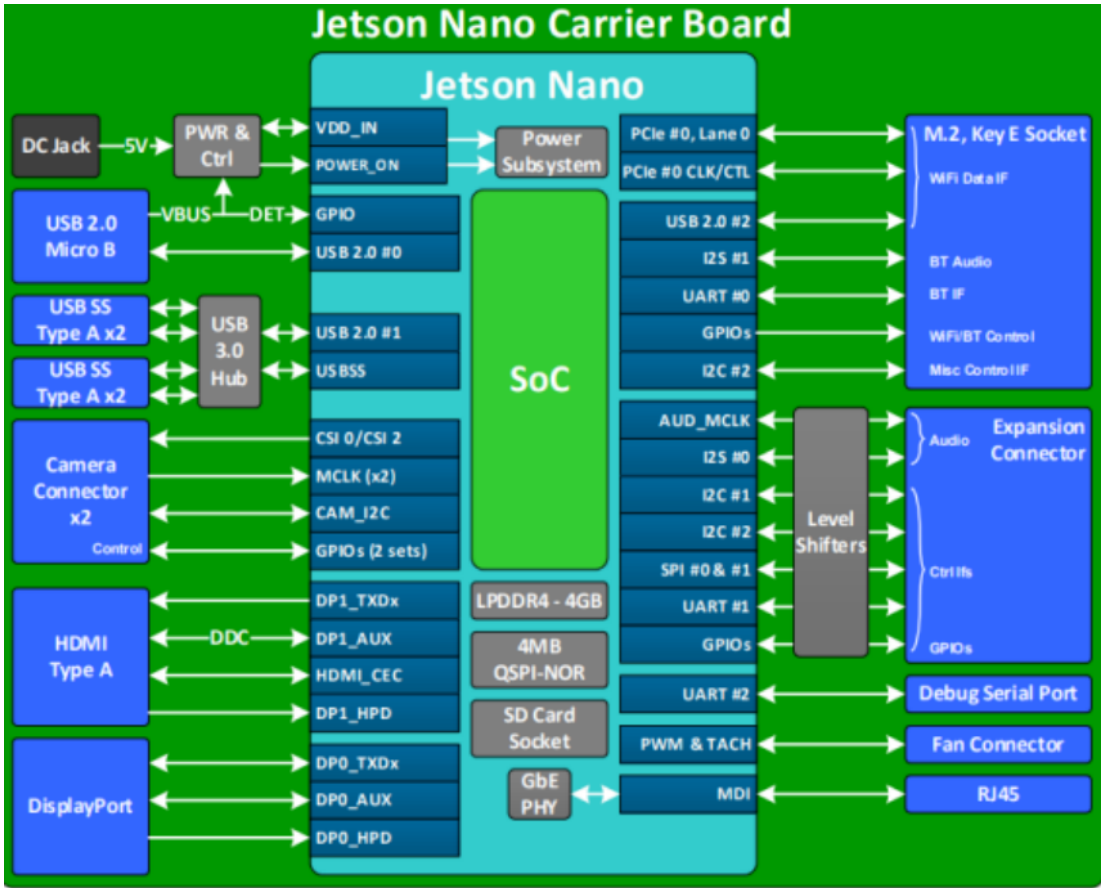


Figura 61 Especificaciones Técnicas de NVIDIA Jetson Nano [45]

Figura 62 . Nvidia Jetson Nano Adquirida

Anexo D. Conexión con la UPS

Anexo E. Cámara instalada

Anexo F. Lugar de aplicacion

Figura 63. Implementación del sistema

Anexo G. Pruebas del sistema en videos pregrabados

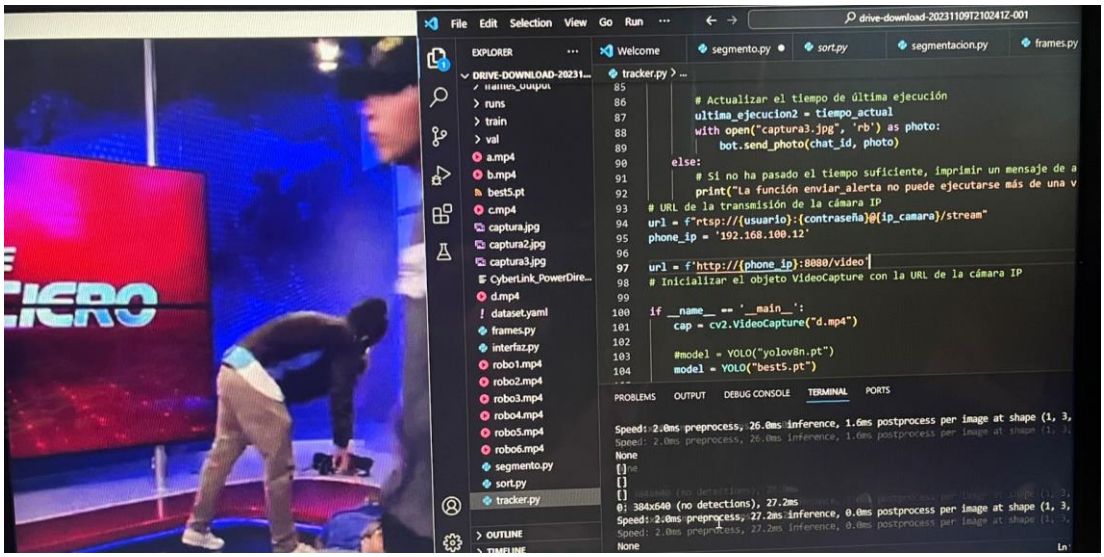


Figura 64. Pruebas del sistema en videos pregrabados 1

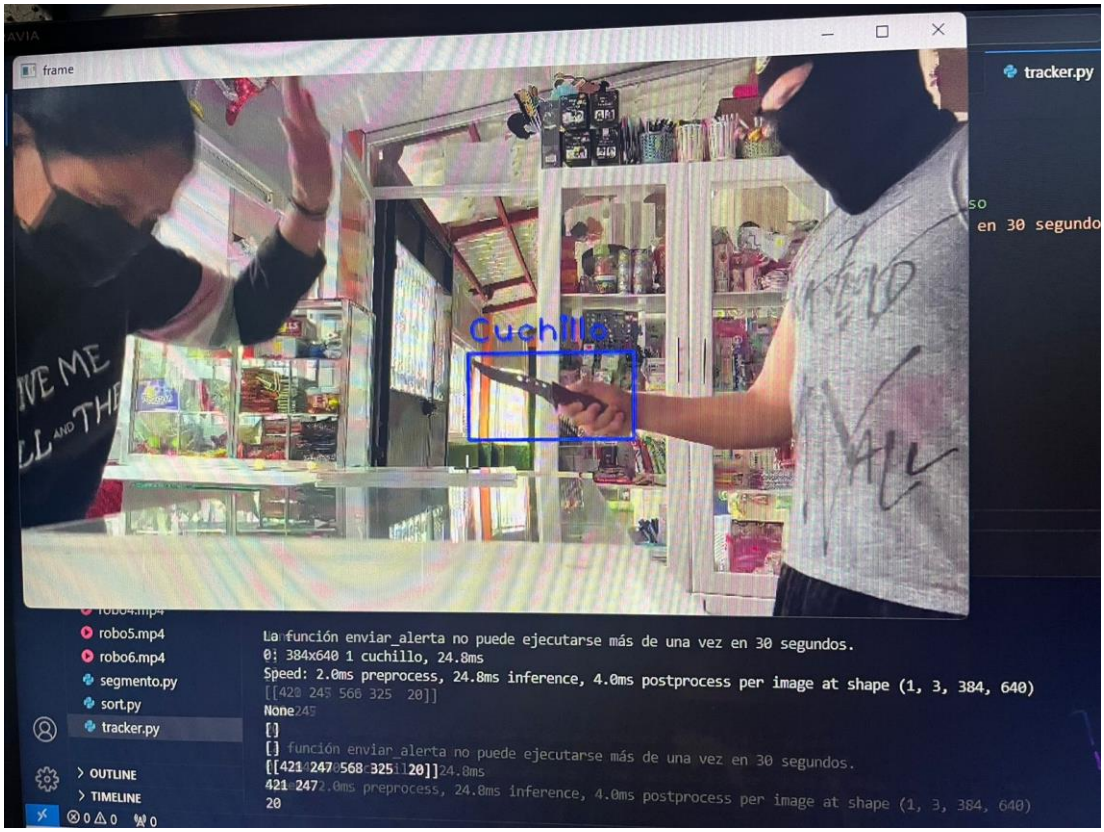


Figura 65. Pruebas del sistema en videos pregrabados 2

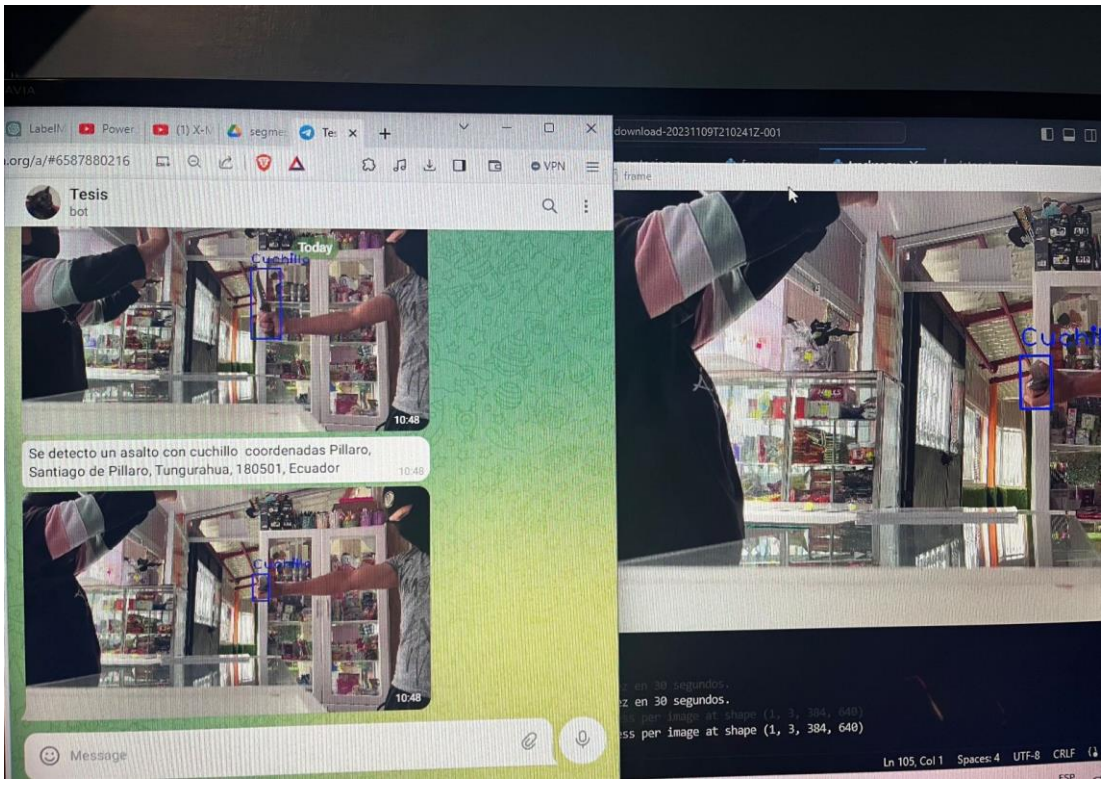


Figura 66. Pruebas del sistema en videos pregrabados 3

Figura 67. Pruebas del sistema en videos pregrabados 4

Anexo H. Pruebas en tiempo real

Figura 68 Pruebas en tiempo real 1

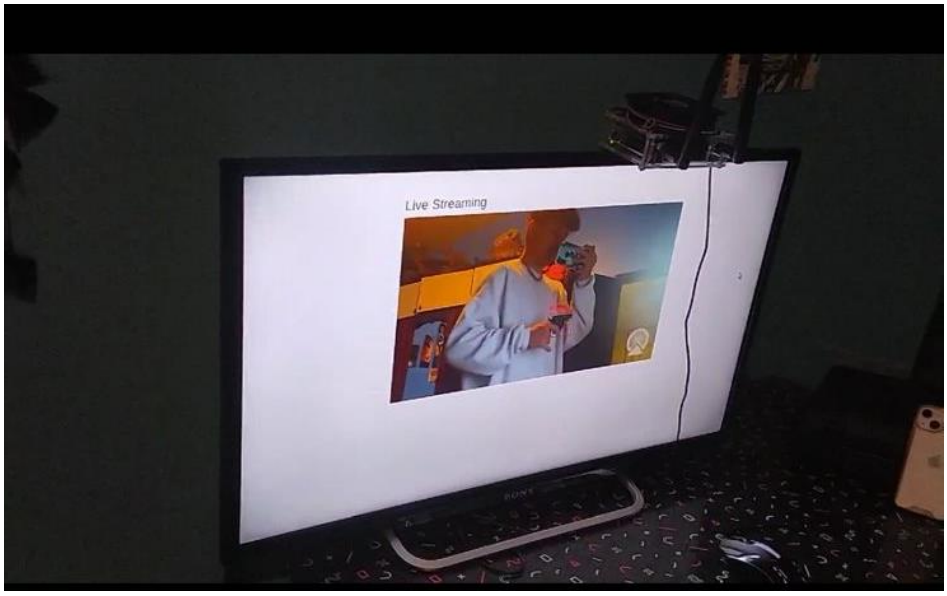


Figura 69 Pruebas en tiempo real 2

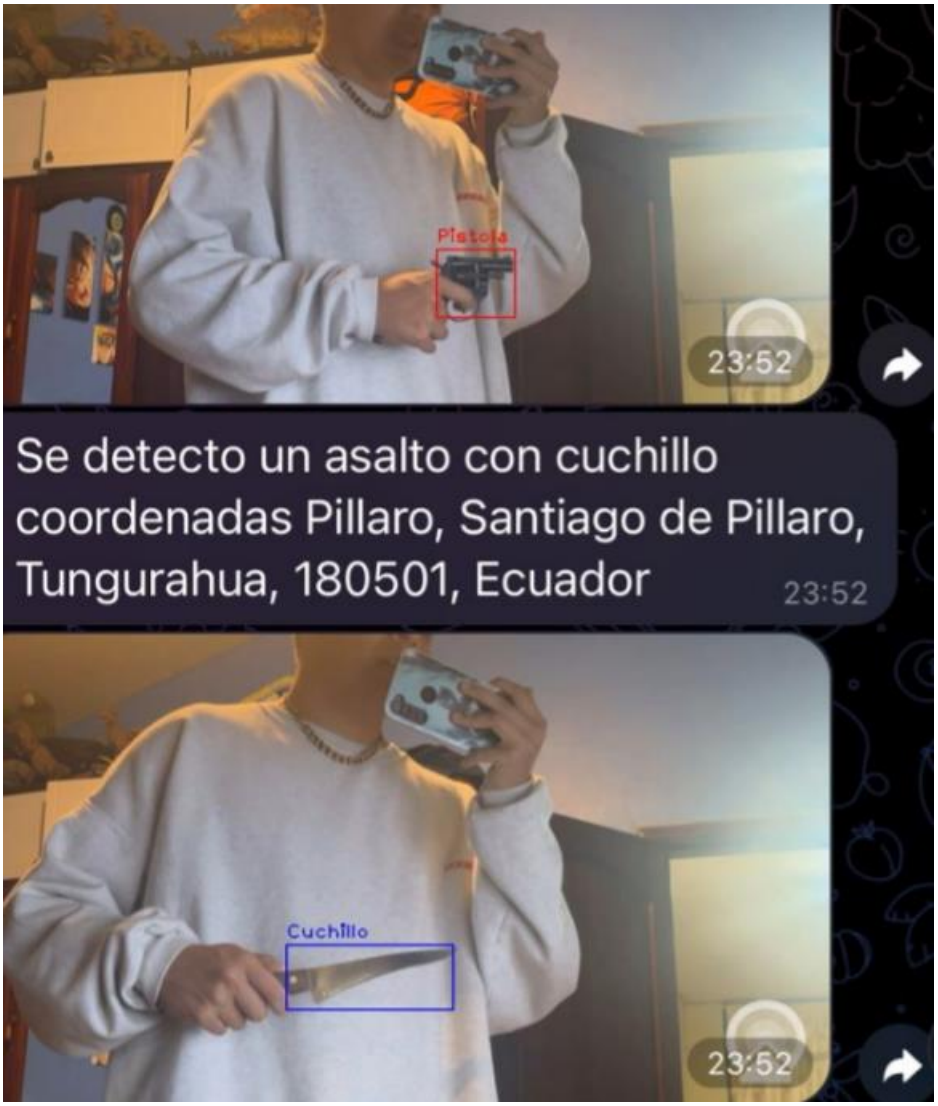


Figura 70 Pruebas en tiempo real 3

Anexo I. Funcionamiento del sistema

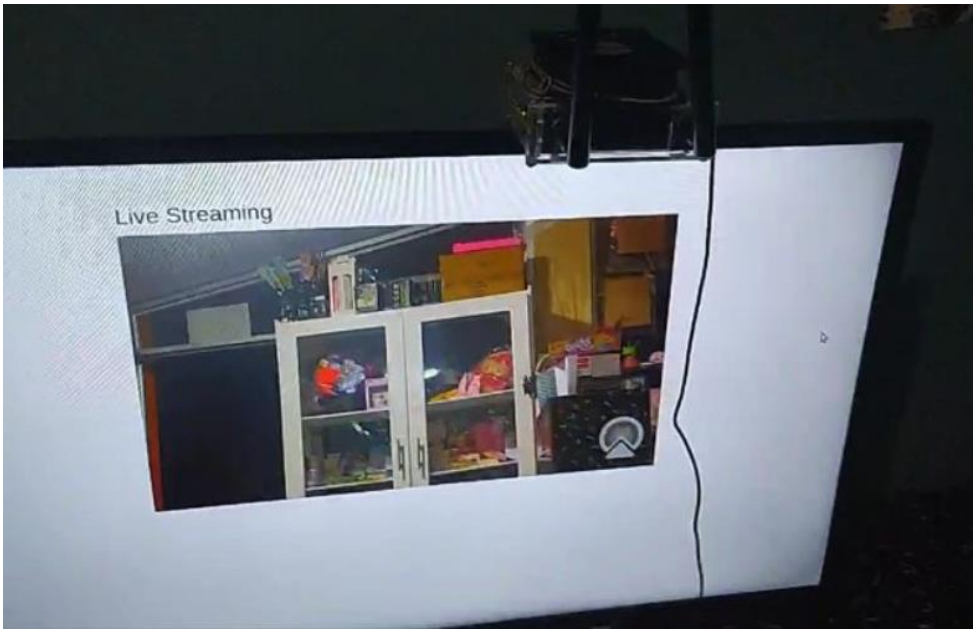


Figura 71. Funcionamiento del sistema 2

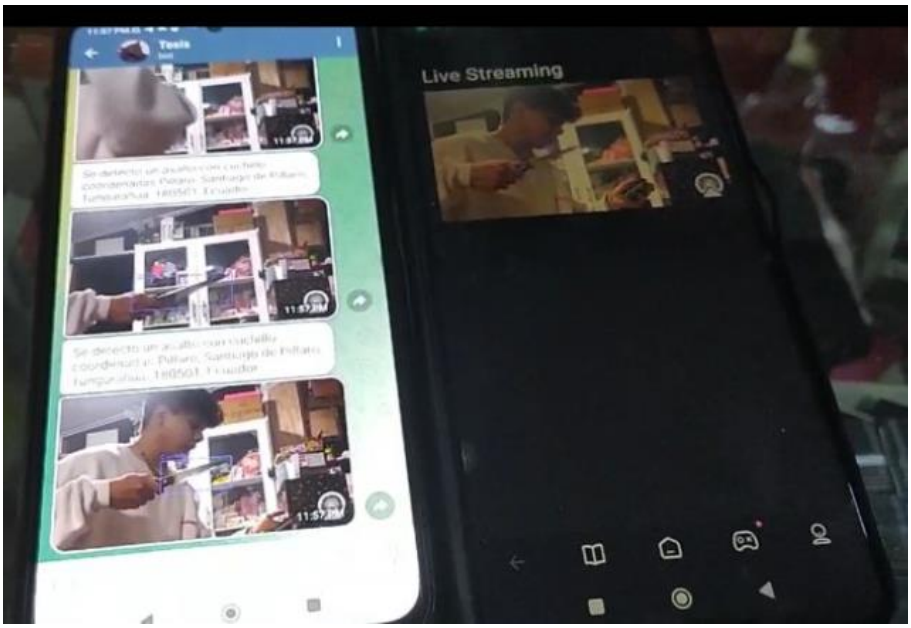


Figura 72. Funcionamiento del sistema 3

Anexo J. Código del sistema

```
import numpy as np
import cv2
from ultralytics import YOLO
from sort import Sort
import itertools
import telebot
from geopy.geocoders import Nominatim
# Dirección IP y detalles de acceso de la cámara
ip_camara = "192.168.100.62"  # Ejemplo de dirección IP
usuario = "admin"
contraseña = "camara22"

# Reemplaza 'TU_TOKEN' con el token que obtuviste de BotFather
TOKEN = '6587880216:AAFEF7bItFes48DFB12z0cniSnflAbjO1no'

# Crea una instancia del bot
bot = telebot.TeleBot(TOKEN)

import time

# Variable para almacenar el último tiempo de ejecución
ultima_ejecucion = 0
ultima_ejecucion2 = 0
def obtener_direccion(latitud, longitud):
    geolocalizador = Nominatim(user_agent="mi_aplicacion")
    ubicacion = geolocalizador.reverse((latitud, longitud))

    if ubicacion:
        print(f"Dirección encontrada: {ubicacion.address}")
        ubicacion2=ubicacion.address
    else:
        print("Dirección no encontrada.")
    return ubicacion2
latitud = -1.1728012
longitud = -78.5498827
direccion = obtener_direccion(latitud, longitud)
direccion= str(direccion)
def enviar_alerta(chat_id, mensaje):
    global ultima_ejecucion

    # Obtener el tiempo actual
    tiempo_actual = time.time()

    # Verificar si ha pasado al menos 30 segundos desde la última ejecución
    if tiempo_actual - ultima_ejecucion >= 10:
        # Si ha pasado el tiempo suficiente, enviar la alerta
        bot.send_message(chat_id, mensaje)

        # Actualizar el tiempo de última ejecución
        ultima_ejecucion = tiempo_actual
        with open("captura.jpg", 'rb') as photo:
            bot.send_photo(chat_id, photo)
    else:
```

```
        # Si no ha pasado el tiempo suficiente, imprimir un mensaje de aviso
        print("La función enviar_alerta no puede ejecutarse más de una vez en 30 segundos.")

def enviar_alerta2(chat_id, mensaje):
    global ultima_ejecucion2

    # Obtener el tiempo actual
    tiempo_actual = time.time()

    # Verificar si ha pasado al menos 30 segundos desde la última ejecución
    if tiempo_actual - ultima_ejecucion2 >= 10:
        # Si ha pasado el tiempo suficiente, enviar la alerta
        bot.send_message(chat_id, mensaje)

        # Actualizar el tiempo de última ejecución
        ultima_ejecucion2 = tiempo_actual
        with open("captura2.jpg", 'rb') as photo:
            bot.send_photo(chat_id, photo)
    else:
        # Si no ha pasado el tiempo suficiente, imprimir un mensaje de aviso
        print("La función enviar_alerta no puede ejecutarse más de una vez en 30 segundos.")
def enviar_alerta3(chat_id, mensaje):
    global ultima_ejecucion2

    # Obtener el tiempo actual
    tiempo_actual = time.time()

    # Verificar si ha pasado al menos 30 segundos desde la última ejecución
    if tiempo_actual - ultima_ejecucion2 >= 10:
        # Si ha pasado el tiempo suficiente, enviar la alerta
        bot.send_message(chat_id, mensaje)

        # Actualizar el tiempo de última ejecución
        ultima_ejecucion2 = tiempo_actual
        with open("captura3.jpg", 'rb') as photo:
            bot.send_photo(chat_id, photo)
    else:
        # Si no ha pasado el tiempo suficiente, imprimir un mensaje de aviso
        print("La función enviar_alerta no puede ejecutarse más de una vez en 30 segundos.")
# URL de la transmisión de la cámara IP
url = f"rtsp://{usuario}:{contraseña}@{ip_camara}/stream"
phone_ip = '192.168.100.12'

url = f'http://{phone_ip}:8080/video'
# Inicializar el objeto VideoCapture con la URL de la cámara IP

if __name__ == '__main__':
    cap = cv2.VideoCapture("robo5.mp4")

    #model = YOLO("yolov8n.pt")
    model = YOLO("best5.pt")

    tracker = Sort()
    tracker2 = Sort()
    tracker3 = Sort()
    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    color = (0, 0, 255)  # Color rojo en formato BGR
    while cap.isOpened():
```

```
        status, frame = cap.read()



        # #frame = cv2.resize(frame, None, fx=0.5, fy=0.5)
        # #************************************************
        # x = int(width * 0.05)  # Coordenada x superior izquierda del rectángulo
        # y = int(height * 0.05)  # Coordenada y superior izquierda del rectángulo
        # width_rect = int(width * 0.9)  # Ancho del rectángulo
        # height_rect = int(height * 0.9)  # Alto del rectángulo

        # thickness = 2  # Grosor del rectángulo
        # cv2.rectangle(frame, (x, y), (x + width_rect, y + height_rect), color, thickness)
        #*********************************************************************
        results = model(frame, stream=True, classes=[0,1,2])#56,8
        #results = model(frame, stream=True)


        for res in results:
            probs = res.probs
            print(probs)
                                    filtered_indices      =      np.where((res.boxes.conf.cpu().numpy()      >
0.30)&(((res.boxes.cls.cpu().numpy() == 1))))[0]
            #filtered_indices2 = np.where(res.boxes.cls.cpu().numpy() == 7)[0]
            #filtered_indices2 = np.where((res.boxes.conf.cpu().numpy() > 0.30))[0]
                                   filtered_indices2      =      np.where((res.boxes.conf.cpu().numpy()      >
0.30)&(((res.boxes.cls.cpu().numpy() == 0))))[0]
                                   filtered_indices3      =      np.where((res.boxes.conf.cpu().numpy()      >
0.10)&(((res.boxes.cls.cpu().numpy() == 2))))[0]
            boxes = res.boxes.xyxy.cpu().numpy()[filtered_indices].astype(int)
            boxes2 = res.boxes.xyxy.cpu().numpy()[filtered_indices2].astype(int)
            boxes3 = res.boxes.xyxy.cpu().numpy()[filtered_indices3].astype(int)
            tracks = tracker.update(boxes)

            tracks = tracks.astype(int)
            tracks2 = tracker2.update(boxes2)
            tracks2 = tracks2.astype(int)
            tracks3 = tracker3.update(boxes3)
            tracks3 = tracks3.astype(int)
            print(tracks)
            print(tracks2)
            print(tracks3)

            for (xmin, ymin, xmax, ymax, track_id) in tracks:
                print(xmin,ymin)
                print(track_id)

                                        cv2.putText(img=frame,     text=f"Pistola",     org=(xmin,     ymin-10),
fontFace=cv2.FONT_HERSHEY_PLAIN, fontScale=2, color=(0,0,255), thickness=2)
                    cv2.rectangle(img=frame,  pt1=(xmin,  ymin),  pt2=(xmax,  ymax),  color=(0,  0,  255),
thickness=2)

                mensaje2 = 'Pistola detectada '+ " coordenadas "+ direccion
                cv2.imwrite("captura.jpg", frame)
                enviar_alerta('1691212127', mensaje2)
```

```
        for (xmin2, ymin2, xmax2, ymax2, track_id2) in tracks2:
            print(xmin2,ymin2)
            print(track_id2)

                    cv2.putText(img=frame, text=f"Situacion de riesgo", org=(xmin2, ymin2-10),
fontFace=cv2.FONT_HERSHEY_PLAIN, fontScale=2, color=(255,0,0), thickness=2)
             cv2.rectangle(img=frame, pt1=(xmin2, ymin2), pt2=(xmax2, ymax2), color=(255, 0, 0),
thickness=2)
            mensaje2 = 'Persona en peligro '+ " coordenadas "+ direccion
            cv2.imwrite("captura2.jpg", frame)
            enviar_alerta2('1691212127', mensaje2)
        for (xmin3, ymin3, xmax3, ymax3, track_id3) in tracks3:
            print(xmin3,ymin3)
            print(track_id3)

                                cv2.putText(img=frame, text=f"Cuchillo", org=(xmin3, ymin3-10),
fontFace=cv2.FONT_HERSHEY_PLAIN, fontScale=2, color=(255,0,0), thickness=2)
             cv2.rectangle(img=frame, pt1=(xmin3, ymin3), pt2=(xmax3, ymax3), color=(255, 0, 0),
thickness=2)
            mensaje2 = 'Se detecto un asalto con cuchillo '+ " coordenadas "+ direccion
            cv2.imwrite("captura3.jpg", frame)
            enviar_alerta3('1691212127', mensaje2)

      # frame = results[0].plot()
   # Comprobar si el marco tiene un tamaño válido antes de mostrarlo
      if frame is not None and frame.shape[0] > 0 and frame.shape[1] > 0:
         cv2.imshow("frame", frame)

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

   cap.release()
```