UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TRABAJO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

"SISTEMA DE DETECCIÓN DE PRESENCIA DE PERSONAS EN EMERGENCIAS PARA LA UNIVERSIDAD TÉCNICA DEL NORTE"

AUTOR: Juan José Vásquez Agudelo

DIRECTOR: Edgar Alberto Maya Olalla

Ibarra-Ecuador

2024

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1004548267		
APELLIDOS Y NOMBRES:	Vásquez Agudelo Juan José		
DIRECCIÓN:	Antonio Cordero y Juan Francisco Bonilla		
EMAIL:	jjvasqueza@utn.edu.ec		
TELÉFONO FIJO:		**TELÉFONO MÓVIL:**	0987677336

DATOS DE LA OBRA	
TÍTULO:	SISTEMA DE DETECCIÓN DE PRESENCIA DE PERSONAS EN EMERGENCIAS PARA LA UNIVERSIDAD TÉCNICA DEL NORTE
AUTOR (ES):	Vásquez Agudelo Juan José
FECHA: DD/MM/AAAA	1/03/2024
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	☒ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	Ingeniero en Telecomunicaciones
ASESOR /DIRECTOR:	Msc. Edgar Alberto Maya Olalla / Edgar Daniel Jarmillo Vinueza

2. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, al día 1 del mes de marzo de 2024

EL AUTOR:

(Firma).....
Nombre: Juan José Vásquez Agudelo

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 29 de Febrero de 2024

Edgar Alberto Maya Olalla
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:
Haber revisado el presente informe final del trabajo de Integración Curricular, el mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

(f) .....
EDGAR ALBERTO MAYA OLALLA
C.C.: ...1002702191

APROBACIÓN DEL COMITÉ CALIFICADOR

El Comité Calificador del trabajo de Integración Curricular "Sistema de detección de presencia de personas en emergencias para la Universidad Técnica del Norte" elaborado por Juan José Vásquez Agudelo, previo a la obtención del título de Ingeniero en Telecomunicaciones, aprueba el presente informe de investigación en nombre de la Universidad Técnica del Norte:

(f):.....
Edgar Alberto Maya Olalla
C.C.:..1002702191.....

(f):.....
Edgar Daniel Jaramillo
C.C.:.....1001545142...

DEDICATORIA

Primero que todo quiero dedicarle esta meta alcanzada a mi familia, quienes han sido mi motor y mi apoyo para motivarme en finalizar este proceso en mi vida

A mi padre y a mi madre quienes me supieron educar y aconsejar en esta etapa tan importante de mi vida la cual me convierte en un profesional.

A mis amigos y compañeros con los que empecé la carrera, y los que fui encontrando en cada tramo, de quienes me llevo grandes recuerdos, porque han sido un pilar fundamental con su apoyo para que juntos terminemos esta etapa académica.

Vasquez Agudelo Juan José

AGRADECIMIENTO

A mis padres que me brindaron todo su apoyo tanto en lo económico, como en mi formación como persona, y que sin su ayuda nada de esto sería posible.

A mis profesores, quienes me compartieron su gran conocimiento para poder solventar problemas que enfrenté durante el camino, gracias a su experiencia y enseñanza.

A mi director de tesis Msc. Edgar Maya y mi asesor de tesis Msc. Edgar Jaramillo por su orientación desde el comienzo de este proyecto, quienes con su conocimiento y experiencia me indicaron como llevar a cabo de la mejor manera mi trabajo de titulación.

Vasquez Agudelo Juan José

RESUMEN EJECUTIVO

En este trabajo de titulación realizado se describe cada paso correspondiente a la metodología utilizada para realizar la implementación del sistema de detección de presencia de personas con el fin de mejorar los tiempos de respuesta en la evacuación y rescate de personas durante una emergencia. Este sistema consiste en un sistema embebido con una cámara la cual toma la foto y mediante el modelo YOLOv7 analiza si hay una persona o no y lo notifica al personal de seguridad por medio de una aplicación móvil. Para diseñar e implementar este sistema se usó la metodología en cascada que consiste en 5 fases. En la fase de requerimientos se investiga de forma técnica como debe ser la arquitectura de hardware y de software del sistema, además de determinar cómo debe ser su funcionamiento.

En la parte de diseño se establecen en base a los requerimientos que software y hardware se va a utilizar para construir e implementar el sistema. Finalmente, en la fase de operación y pruebas se compara el rendimiento de 3 modelos (YOLOv5s, YOLOv5x y YOLOv7), que fueron entrenados con el fin de elegir cual se adapta de mejor manera a las necesidades del sistema, y comprobar que tal funciona el modelo electo mediante la implementación del sistema dentro del aula de clases. En donde el modelo que obtuvo un mejor rendimiento con un 96% de precisión es el modelo YOLOv7, ya que, al ser un modelo bastante optimizado en comparación de los otros dos, tiene un balance entre precisión y rendimiento cuanto a tiempo se refiere, por lo hace ideal para este proyecto.

Palabras clave: emergencia, YOLO, arquitectura, entrenamiento, precisión, visión artificial.

ABSTRACT

In this completed thesis work, each step of the methodology used for the implementation of a people presence detection system is described. The goal is to improve response times in the evacuation and rescue of individuals during emergencies. This system involves an embedded setup with a camera that captures images. Using the YOLOv7 model, it analyzes whether a person is present or not, notifying security personnel through a mobile application. The design and implementation of this system followed the cascade methodology, comprising five phases. The requirements phase involved technical investigation into the hardware and software architecture, as well as determining the system's functionality.

In the design phase, decisions are made based on the requirements regarding the software and hardware that will be used to build and implement the system. Finally, in the operation and testing phase, the performance of three models (YOLOv5s, YOLOv5x, and YOLOv7) is compared. These models were trained to determine which one best suits the system's needs. The selected model's functionality is verified through the system's implementation in a classroom setting. The YOLOv7 model outperformed the others with 96% accuracy, as it is highly optimized compared to the other two, striking a balance between precision and performance in terms of time, making it ideal for this project.

Keywords: emergency, YOLO, architecture, training, precision, computer vision.

LISTA DE SIGLAS

DSGR. Departamento de Seguridad y Gestión de Riesgos.

SBC. Single Board Computer.

IA. Inteligencia Artificial.

CBIR. Content Based Image Retrieval.

OCR. Optical Character Recognition.

YOLO. You Only Look Once.

RCNN. Region-based Convolutional Neural Network.

API. Aplication Programming Interface.

FPS. Frames Per Second.

CNN. Convolutional Neural Network.

COCO. Common Objects in Context.

mAP. mean Average Precision.

NMS. Non Maximum Suppresion.

IoU. Intersection over Union.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En este capítulo se abordarán los parámetros iniciales con los que se plantea el proyecto y su desarrollo, comprendiendo el tema, problema, objetivos, alcance y justificación, estableciendo así, el procedimiento a realizar a lo largo del tiempo que requiera el proyecto en cuestión.

1.1. Problema de investigación

La Universidad Técnica del Norte cuenta con un total aproximado de 12000 personas dentro de las cuales se encuentran estudiantes, docentes y administrativos tanto de jornada diurna como nocturna (Universidad Técnica del Norte, 2022). Diariamente se tiene una afluencia aproximada de 7000 personas diarias dentro de la jornada laboral de 7h00 am hasta las 20h00 pm además de un aproximado de 500 personas visitantes por día. La Universidad Técnica del Norte posee de parte del Departamento de Seguridad y Gestión de Riesgos un Plan de Emergencia con base a las políticas de Gestión de Riesgos, Seguridad Industrial y Salud Ocupacional donde se plantea como objetivo el tener a consideración un plan de acción que se ejecute de manera oportuna ante cualquier situación de emergencia tales como: incendios, explosiones, amenazas de bomba, violencia civil, movimientos telúricos, etc. (PLAN DE EMERGENCIAS UTN, 2021).

En caso de una catástrofe el plan de emergencias es ejecutado en base al tipo de emergencia, comenzando por la detección humana por parte de cada uno de los que conforman el personal de seguridad, luego se realiza la activación de la alarma, en caso

de existir ausencia por parte del Centro de Mando y Control actuarán los guardias, se procede a evaluar cuál es la situación de la emergencia, dependiendo de si es una falsa alarma o emergencia de grado I se realiza un informe para seguridad integral de contrario, siendo una emergencia tipo II o III se realiza el contacto a organismos externos, se controla la emergencia evacuando a todos los ocupantes de cada edificio del campus, y finalmente se emiten informes a seguridad integral. En la Universidad Técnica del Norte se tiene una propuesta de un sistema de alerta temprana para movimientos telúricos, pero no aporta con una solución para la detección de presencia en caso de una emergencia, ya que Según el director del DSGR Edwar Vásquez, necesitan fortalecer las capacidades de respuesta ante una emergencia.

En base a la necesidad de un sistema de detección de personas se pretende que con la implementación de este sistema experto se solucionen los inconvenientes mencionados anteriormente teniendo un monitoreo en tiempo real para detección de presencia mediante sensores, los cuales enviarán de manera inalámbrica los datos obtenidos hacia una aplicación móvil que tendrán los encargados de emergencias de cada edificio para que el COE institucional optimice los tiempos y actúe en base a estos datos ejecutando los protocolos a seguir dentro del plan de emergencias de la UTN.

1.2. Justificación

Dentro del plan de emergencias de la Universidad Técnica del Norte se afirma que cada una de las instalaciones, muy aparte de su uso, son susceptibles constantemente a amenazas de cualquier índole afectando a los ocupantes dentro del lugar donde se encuentren como incendios, explosiones, inundaciones, sismos, etc. (PLAN DE EMERGENCIAS UTN, 2021).

El COE institucional establece que la decisión de evacuación será tomada por parte del Centro de Mando y Control la cuál será declarada por el Coordinador General de Emergencias, donde se suspenderán todo tipo de actividades que se estén realizando dentro de la universidad, el Coordinador General de Emergencias analizará el tipo de emergencia para establecer si se retoman o no las actividades posteriormente (PLAN DE EMERGENCIAS UTN, 2021).

La finalidad del proyecto es aportar a la Universidad Técnica del Norte con la implementación de un sistema que permita detectar la presencia de alguna persona que se quede adentro de las instalaciones ya sea por pánico al intentar evacuar como en el peor de los casos se quede atrapado debido a algún incendio, derrumbe o cualquier tipo de emergencia nombrado dentro del plan de emergencias de la UTN, permitiendo que el tiempo en la ejecución de los protocolos de rescate del DSGR sea optimizado pudiendo evitar algún deceso, realizando un rescate a tiempo ya que es más demoroso buscar aula por aula si existe alguna persona atrapada dentro de las instalaciones, de igual manera mediante la aplicación móvil se reporte al DSGR en caso de presentarse una emergencia fuera del horario laboral ya que no existe un monitoreo constante las 24 horas del día.

1.3. Objetivos

1.3.1. Objetivo General

Implementar un sistema embebido aplicando visión artificial para la detección de personas, una completa evacuación y rescate dentro de las instalaciones de la Universidad Técnica del Norte.

1.3.2. Objetivos Específicos

- Realizar el estado del arte sobre emergencias, y temas correspondientes a cada capa de la arquitectura del sistema.
- Diseñar el hardware y software tomando en cuenta los requerimientos correspondientes a cada capa de la arquitectura para la detección de presencia.
- Construir el sistema embebido entendiendo las características funcionales del sistema para poder implementarlo dentro de la Universidad.
- Probar el prototipo del sistema de detección de presencia para validar resultados en cuanto a la detección de personas y a una evacuación completa del personal.

CAPÍTULO II

MARCO TEÓRICO

En este capítulo se tratarán cada uno de los temas de interés que se relacionan al tema del proyecto, como lo son las emergencias, y cada parte de la arquitectura del sistema de detección de presencia de personas para la Universidad Técnica del Norte.

2.1. Emergencia

Una emergencia es una situación que requiere de atención inmediata porque implica un riesgo o un daño para la vida o la salud de las personas. Por ejemplo, si alguien sufre un accidente grave, un incendio o un desastre natural, se enfrenta a una emergencia y necesita ayuda médica o de rescate. Una emergencia puede ser causada por factores externos o internos, y puede afectar a una persona, a un grupo o a una comunidad entera. Lo importante es actuar con rapidez y eficacia para evitar o minimizar las consecuencias negativas de la emergencia (Etecé, 2021).

Es una situación que requiere una respuesta rápida y eficaz para evitar o minimizar un daño mayor. Una emergencia puede ser de diferente naturaleza, como natural, social, ambiental, sanitaria o personal. Lo que caracteriza a una emergencia es que implica un riesgo para la vida, la salud, la seguridad o el bienestar de las personas o de los bienes materiales. Por lo tanto, una emergencia exige una acción inmediata y coordinada de los agentes involucrados, como los servicios de emergencia, las autoridades, las organizaciones humanitarias y la población afectada (Gobierno de México, 2023).

2.2. Grados de una emergencia

Según el plan de emergencias de la UTN (2021) existen tres grados de la emergencia:

"…

1. *Emergencia en Fase inicial o Conato (Grado I).*
2. *Emergencia Parcial (Grado II).*
3. *Emergencia General (Grado III).*

- ***Emergencia en Fase inicial o Conato (Grado I).***

Situación de riesgo que altera a las personas del sitio; puede tener su origen en un incendio o en otra situación de peligro de pequeña escala. Este tipo de situaciones interrumpe el normal funcionamiento de las actividades de operación, pero puede ser controlada con los recursos disponibles por funcionarios, estudiantes y visitantes del sitio ya que puede ser solucionada por la primera respuesta.

- ***Emergencia Parcial (Grado II).***

Situación de riesgo causada por un fuego o un suceso desfavorable de mediana escala; por sus condiciones necesita otros recursos como: colaboración interna, colaboración externa, colaboración de Seguridad Física, Seguridad y Salud Ocupacional, ya que no puede ser controlada de inmediato como un conato.

- ***Emergencia General (Grado III).***

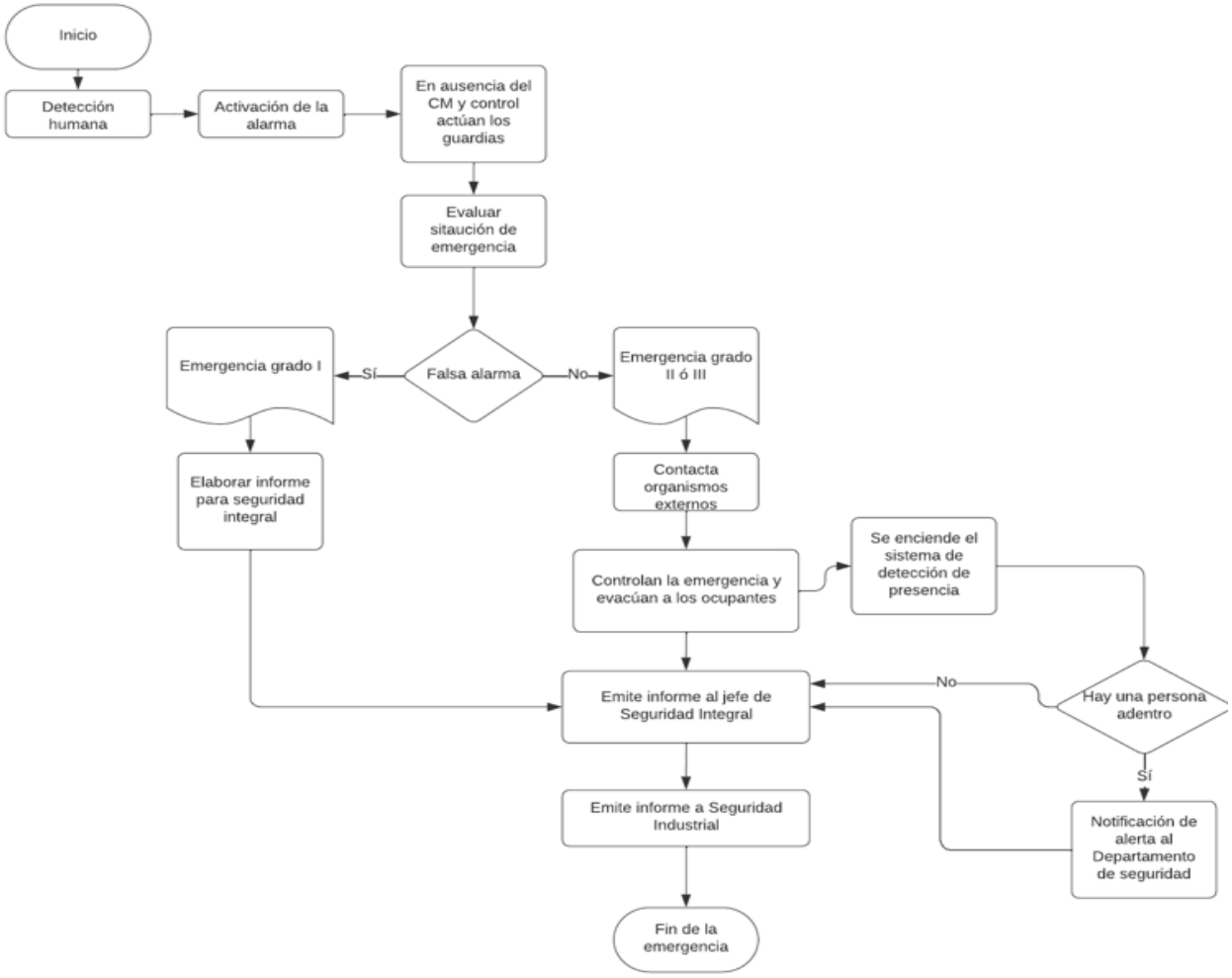
Situación de riesgo que por sus condiciones, tamaño y consecuencias exige pedir la participación inmediata, amplia y completa de los recursos internos y externos…"

2.2.1. Actuación frente a emergencias del personal de seguridad

Dentro del plan de emergencias de la Universidad Técnica del Norte, se comprenden los pasos que el personal de seguridad debe seguir y se comprenden de mejor manera en el siguiente diagrama de flujo, tomando en cuenta como proceso adicional el sistema de detección de presencia:

Figura 1

Diagrama de flujo que detalla el procedimiento que debe seguir el personal de seguridad ante una emergencia



Este diagrama de flujo comprende los procesos a seguir por parte del personal de seguridad en caso de que se presente una emergencia, en conjunto con el funcionamiento del sistema de detección de presencia.

2.3. Sistemas embebidos

Los sistemas embebidos son sistemas de computación diseñados con el propósito de llevar a cabo una tarea particular. Normalmente, se encuentran integrados dentro de dispositivos físicos, tales como vehículos, smartphones, reproductores de música y

electrodomésticos. Estos sistemas embebidos están experimentando un crecimiento en su complejidad y capacidad, siendo utilizados en diversas aplicaciones de manera amplia.

Estos sistemas están constituidos por la integración de elementos físicos y programas informáticos. Se destacan por su reducido tamaño, su capacidad de operar de manera autónoma y sin necesidad de intervención humana. Además, poseen la habilidad de resistir eventos adversos y reiniciarse de manera automática (Luchetti, 2021).

2.3.1. Componentes de un sistema embebido

Según (J.L, 2023) los sistemas embebidos se componen de 3 partes principales:

"…

- ***Hardware:*** *El conjunto de elementos físicos que llevan a cabo las funciones del sistema constituye el hardware de un sistema embebido. Esta composición generalmente comprende un microcontrolador, sensores, actuadores y memoria.*
- ***Software y Firmware:*** *La parte lógica de un sistema embebido engloba el firmware o programas permanentes, los controladores de dispositivos y las aplicaciones diseñadas específicamente para cumplir con las funciones del sistema.*
- ***Sistema operativo:*** *El sistema operativo de un sistema embebido es el programa informático que administra los recursos del hardware y ejecuta las aplicaciones. Por lo general, se trata de un sistema operativo en tiempo real, lo que asegura la ejecución puntual de las aplicaciones*…"

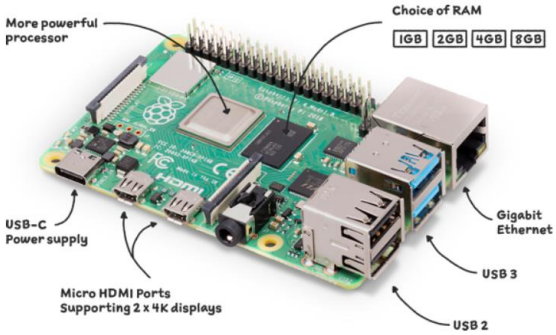
2.3.2. Raspberry Pi 4 modelo B

La Raspberry Pi 4 Model B es un dispositivo de placa única (SBC) asequible diseñado con la intención de fomentar la educación en informática y programación. Esta

placa tiene la versatilidad de operar como una computadora personal completa, permitiendo actividades como la navegación web, la reproducción de videos, la creación de documentos, la programación y jugar (AV Electronics, s. f.).

Figura 2

Raspberry Pi 4 modelo B



Nota. Adaptado de (Raspberry Pi, s. f.-a)

- **Especificaciones técnicas del Raspberry Pi 4 modelo B**

Tabla 1

Especificaciones técnicas del Raspberry Pi 4 modelo B

Especificación	**Detalles**
Procesador	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz
Memoria RAM	4GB
Conectividad Inalámbrica	2.4 GHz y 5.0 GHz IEEE 802.11ac, Bluetooth 5.0, BLE
Conectividad por Cable	Gigabit Ethernet

Puertos USB	2 puertos USB 3.0; 2 puertos USB 2.0
Encabezado GPIO Raspberry Pi estándar	40 pines (totalmente compatible con placas anteriores)
Salidas de Video	2 × micro-HDMI® (soporta hasta 4kp60)
Puerto de Pantalla MIPI DSI de 2 carriles	2-lane MIPI DSI
Puerto de Cámara MIPI CSI de 2 carriles	2-lane MIPI CSI
Salida de Audio y Video estéreo de 4 polos	4-pole stereo audio y composite video
Capacidad de Video (Decodificación/ Codificación)	H.265 (decodificación 4kp60), H.264 (decodificación 1080p60, codificación 1080p30)
Gráficos	OpenGL ES 3.1, Vulkan 1.0
Ranura para Tarjeta Micro-SD	Para cargar el sistema operativo y almacenar datos
Alimentación	5V DC a través del conector USB-C (mínimo 3A*) o mediante el encabezado GPIO (mínimo 3A*)
Alimentación a través de Ethernet (PoE) habilitada	(requiere un adaptador PoE por separado)
Temperatura de Operación	0 - 50 grados Celsius en ambiente

Nota. Adaptado de (Raspberry Pi, s. f.-b)

2.3.3. Proyectos usando el Raspberry Pi 4 modelo B aplicado a la visión artificial

El Raspberry Pi 4 modelo B tiene un sin número de proyectos realizados en cuanto a la visión artificial se refiere. La siguiente tabla detalla 5 proyectos que fueron realizados con este sistema embebido, contemplando parámetros como librerías utilizadas, lenguaje de programación, cámara que se utilizó, y los nombres de los autores que realizaron cada uno de los proyectos.

Tabla 2

Proyectos de visión artificial usando la Raspberry Pi 4 modelo B

Tema	**Librerías**	**Lenguaje de programación**	**Cámara**	**Autor**
Sistema seleccionador de botellas mediante visión artificial en Raspberry Pi	OpenCV, Tkinter	Python	acA640	Jhulius Castillo Suárez
Desarrollo de un sistema para detectar los colores en tejidos con hilo utilizando visión artificial	OpenCV, Numpy	Python	Arducam M12	Osejos Rocha Santiago Martín
Detección de somnolencia utilizando técnicas de visión artificial en entornos móviles.	OpenCV	Python	OV5647	Macarena Quiroga, Emilio Melo
Automatización de un sistema identificador y posicionador de objetos a	OpenCV	Python	Logitech C270	Anthony Stalin Llerena Buenaño, Marc

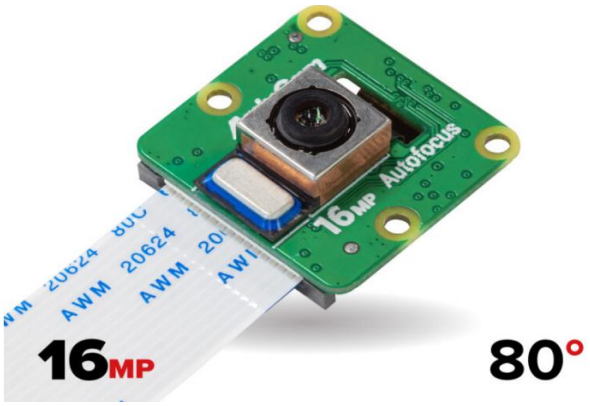
través de un brazo robótico mediante visión artificial con lenguaje python				Jens Salazar Villamar
Diseño de un sistema de detección del vector Aedes Aegypti utilizando visión artificial para la fumigación del pabellón N◦4	OpenCV, Tensor Flow	Python	Ebic WC-1080	Huanambal Esquén Diana Solansh, Timaná Ramos Luis Miguel

2.3.4. Arducam IMX519

Es una placa de cámara de 16MP de alta resolución con enfoque automático, diseñada para aprovechar al máximo las capacidades de Raspberry Pi V1, V2 y HQ, sin comprometer las especificaciones. Ofrece mejoras significativas y es ideal para diversas aplicaciones industriales y de consumo. Es compatible con todos los modelos de Raspberry Pi, funciona con el último software de la fundación y utiliza algoritmos de ajuste similares a los módulos de cámara oficiales (Arducam, s. f.).

Figura 3

Arducam IMX519



Nota. Adaptado de (Arducam, s. f.)

Dentro de la página oficial de Arducam se contemplan las especificaciones técnicas de la cámara Arducam IMX519, tal cual se puede apreciar en la Tabla 3.

Tabla 3

Especificaciones técnicas de la cámara Arducam IMX519

Característica	Valor
Sensor	Sony IMX519
Resolución	16 megapíxeles
Tamaño óptico	1/2.53"
Distancia focal	4.28 mm
Relación focal (F-Stop)	1.75
Modos de vídeo	1080p30, 720p60
Integración con Linux	Controlador V4L2 disponible

Nota. Adaptado de (Arducam, s. f.)

2.4. Visión artificial

La visión artificial o visión por computadora se refiere a la disciplina científica y tecnológica que capacita a las máquinas para observar, analizar imágenes digitales, realizar tareas específicas o comprender el contexto de la escena visual (García, 2012).

La visión artificial, un subcampo de la inteligencia artificial (IA), capacita a las computadoras y sistemas para extraer información significativa de imágenes digitales, vídeos y otros datos visuales, y posteriormente tomar acciones o proporcionar recomendaciones basadas en esa información. Mientras que la IA permite a las computadoras pensar, la visión artificial les permite adquirir la capacidad de ver, observar y comprender (IBM, s. f.).

IBM dice que:

"... la visión artificial opera de manera análoga a la percepción visual humana, aunque con una desventaja inicial. Los seres humanos cuentan con la ventaja de toda una vida de experiencia para aprender a distinguir objetos, evaluar distancias, detectar movimientos y percibir cualquier anomalía en una imagen."

2.4.1. Funcionamiento de la visión artificial

Los sistemas de visión artificial emplean la tecnología de inteligencia artificial (IA) para emular las funciones del cerebro humano encargadas de reconocer y categorizar objetos. Los expertos en informática adiestran las computadoras para que sean capaces de reconocer datos visuales mediante la entrada de grandes volúmenes de información. Los algoritmos de aprendizaje automático (ML) identifican patrones recurrentes en estas imágenes o videos y aplican ese conocimiento para identificar con precisión imágenes desconocidas (AWS, 2023). La visión artificial puede realizar distintos trabajos que pueden ser implementados dentro de un sistema como pueden ser:

- ***Clasificación de imágenes.***

  Dentro de las responsabilidades de la visión por computadora, la clasificación de imágenes se destaca debido a su función fundamental en la tecnología moderna. En esencia, consiste en asignar una etiqueta o descripción a una imagen completa basándose en datos previos de entrenamiento que provienen de imágenes previamente etiquetadas. Aunque este procedimiento puede parecer sencillo a simple vista, en realidad implica un minucioso análisis de la imagen a nivel de píxeles con el propósito

de identificar la etiqueta más apropiada para la imagen en su conjunto. Esta actividad nos provee de datos e información valiosa, permitiéndonos tomar decisiones fundamentadas y lograr resultados prácticos (SuperAnnotate, 2023).

- **Detección de objetos.**

Se trata de una tecnología de aprendizaje profundo que permite la detección de elementos, individuos, estructuras y vehículos en imágenes y videos. El proceso radica en la identificación de estos objetos por medio de un marco delimitador en la imagen, y en cuanto a la clasificación de imágenes, se reduce a la capacidad de categorizar si un objeto se encuentra presente en la imagen o no, evaluando su probabilidad (Patel, 2020).

- **Seguimiento de objetos.**

El seguimiento de objetos, que se basa en el aprendizaje profundo, implica la toma de un grupo inicial de detecciones de objetos, generando una identificación exclusiva para cada una de estas detecciones iniciales, y luego monitoreando el movimiento de los objetos detectados a medida que atraviesan los cuadros de un video. Dicho de otro modo, el seguimiento de objetos consiste en la capacidad de identificar de manera automática los objetos en un video y comprenderlos como un conjunto de rutas con una gran precisión (Klingler, 2023).

- **Segmentación.**

La segmentación de imágenes implica la partición de una imagen en diversas regiones u elementos coherentes y de relevancia, basándose en sus propias cualidades intrínsecas, como el matiz, la textura, la forma o el brillo. Su propósito radica en simplificar o modificar la representación de una imagen de modo que sea más significativa y fácil de examinar. En este procedimiento, cada píxel recibe una

identificación de tal manera que aquellos píxeles pertenecientes a una misma categoría comparten una identificación común (Acharya, 2022).

- **Recuperación de imágenes basada en el contenido.**

La Recuperación de Imágenes Basada en el Contenido (CBIR, por sus siglas en inglés) es una forma de recuperar imágenes de una base de datos. En CBIR, un usuario especifica una imagen de consulta y obtiene las imágenes en la base de datos similares a la imagen de consulta. Para encontrar las imágenes más similares, CBIR compara el contenido de la imagen de entrada con las imágenes de la base de datos. Más específicamente, CBIR compara características visuales como formas, colores, texturas e información espacial y mide la similitud entre la imagen de consulta y las imágenes en la base de datos con respecto a esas características (baeldung, 2022).

2.4.2. Procesos de la visión artificial

Al hablar de la visión artificial y comprendiendo de manera teórica cómo funciona, todo ese proceso de subdivide en 6 procesos como se muestra en la Tabla 4.

Tabla 4

Procesos de la visión artificial

Procesos	**Nivel de visión**	**Entrada**	**Salida**	**Área**
Captura	Bajo	Imagen	Imagen	Procesamiento de imágenes
Preprocesamiento	Bajo	Imagen	Imagen	Procesamiento de imágenes

Segmentación	Medio	Imagen	Grupos de píxeles en bruto	Análisis de imágenes
Descripción	Medio	Objetos o regiones	Información cuantitativa de los objetos o regiones	Análisis de imágenes
Reconocimiento (clasificación)	Medio	Información cuantitativa	Objetos clasificados en categorías	Análisis de imágenes
Interpretación	Alto	Objetos clasificados en categorías	Compresión de la escena	Visión por computador

Nota. Adaptado de (S & S, 2015).

La tabla describe un proceso fundamental en el procesamiento de imágenes digitales, que consta de varias etapas las cuales fueron mencionadas anteriormente.

Es fundamental señalar que, por lo general, estos procedimientos siguen un orden secuencial, en el que los resultados de una etapa alimentan directamente la siguiente. La selección de los procesos a emplear se basa en la complejidad del problema en cuestión, y no todas las fases son obligatorias en todas las situaciones.

2.4.3. Librerías de visión artificial

Las bibliotecas de visión por computadora son herramientas y estructuras de software que proporcionan una variedad de capacidades para analizar, procesar y comprender información visual, como imágenes y videos. Estas bibliotecas se han desarrollado para asistir en diversas tareas de visión por computadora, que incluyen desde la manipulación de imágenes hasta la identificación de objetos, la división de imágenes, el reconocimiento facial, el reconocimiento óptico de caracteres (OCR) y más. En esencia,

estas bibliotecas simplifican la complejidad de procesar información visual, lo que facilita a desarrolladores e investigadores la creación de aplicaciones y modelos de visión por computadora de manera más eficiente (K, 2023).

- **OpenCV.**

  Una biblioteca de código abierto que contiene implementaciones que abarcan más de 2500 algoritmos. Además, está especializada en el sistema de visión artificial y machine learning (Rodríguez, 2021).

Tabla 5

Versiones de OpenCV y sus requerimientos mínimos

Versión	Sistema operativo	Lenguaje de programación	Dependencias
4.5.4	Windows, macOS, Linux, Android, iOS	C/C++, Python, Java	C++11, CMake, CUDA (opcional), OpenCL (opcional), Python 3.6+, NumPy 1.16+, Java 8+
4.4.0	Windows, macOS, Linux, Android, iOS	C/C++, Python, Java	C++11, CMake, CUDA (opcional), OpenCL (opcional), Python 3.5+, NumPy 1.13+, Java 8+
4.3.0	Windows, macOS, Linux, Android, iOS	C/C++, Python, Java	C++11, CMake, CUDA (opcional), OpenCL (opcional), Python 3.4+, NumPy 1.12+, Java 8+
4.2.0	Windows, macOS, Linux, Android, iOS	C/C++, Python, Java	C++11, CMake, CUDA (opcional), OpenCL (opcional), Python 3.3+, NumPy 1.11+, Java 8+
4.1.0	Windows, macOS, Linux, Android, iOS	C/C++, Python, Java	C++11, CMake, CUDA (opcional), OpenCL (opcional), Python 3.2+, NumPy 1.10+, Java 8+

4.0.0	Windows, macOS, Linux, Android, iOS	C/C++, Python, Java	C++11, CMake, CUDA (opcional), OpenCL (opcional), Python 3.1+, NumPy 1.9+, Java 8+
3.4.2	Windows, macOS, Linux, Android, iOS	C/C++, Python, Java	C++11, CMake, CUDA (opcional), OpenCL (opcional), Python 2.7+, NumPy 1.8+, Java 7+
3.3.1	Windows, macOS, Linux, Android, iOS	C/C++, Python, Java	C++11, CMake, CUDA (opcional), OpenCL (opcional), Python 2.7+, NumPy 1.7+, Java 7+
3.2.0	Windows, macOS, Linux, Android, iOS	C/C++, Python, Java	C++11, CMake, CUDA (opcional), OpenCL (opcional), Python 2.7+, NumPy 1.6+, Java 7+
3.1.0	Windows, macOS, Linux, Android, iOS	C/C++, Python, Java	C++11, CMake, CUDA (opcional), OpenCL (opcional), Python 2.7+, NumPy 1.5+, Java 7+
3.0.0	Windows, macOS, Linux, Android, iOS	C/C++, Python, Java	C++11, CMake, CUDA (opcional), OpenCL (opcional), Python 2.7+, NumPy 1.4+, Java 7+

- **Pytorch.**

Se refiere a una biblioteca de código abierto concebida con el enfoque en Python y diseñada para proyectos de aprendizaje automático. Su enfoque principal se centra en la diferenciación automática, cálculos de tensores y aprovechamiento de la aceleración de GPU. Esta característica lo convierte especialmente adecuado para aplicaciones avanzadas de aprendizaje automático, como el aprendizaje profundo. PyTorch ha ganado gran popularidad entre los investigadores gracias a la flexibilidad que ofrece con Python. La creación de capas de datos personalizadas y la construcción

de arquitecturas de red resultan particularmente sencillas mediante el uso de Python (Oracle, 2022).

Tabla 6

Versiones de Pythorch y sus requerimientos

Versión	Sistema Operativo	Lenguaje de programación	Dependencias
main (unstable)	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v2.2.0 (release candidate)	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v2.1.0 (stable release)	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v2.0.0	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v1.13	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v1.12	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)

v1.11	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v1.10	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v1.9.1	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v1.9.0	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v1.8.1	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v1.8.0	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v1.7.1	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v1.7.0	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)

v1.6.0	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v1.5.1	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v1.5.0	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v1.4.0	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v1.3.1	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)
v1.3.0	Linux, macOS, Windows	Python	NumPy, SciPy, Matplotlib, CUDA (opcional)

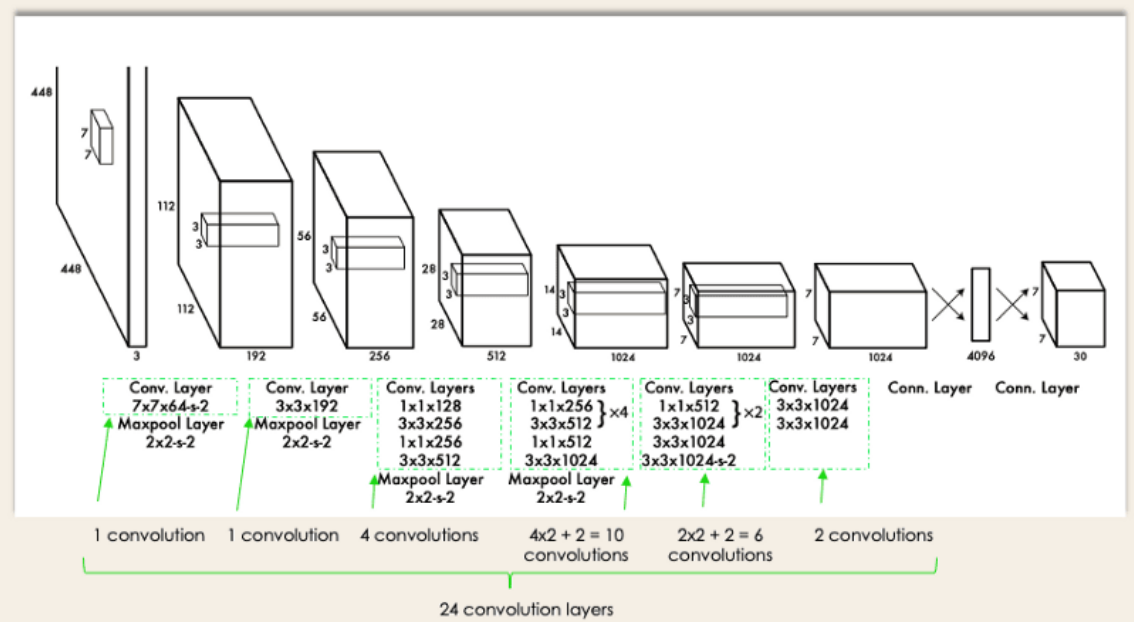
2.5. Modelo YOLO

La propuesta de YOLO (You Only Look Once) introduce una red neuronal integral que simultáneamente realiza predicciones de cajas delimitadoras y probabilidades de clases, marcando un cambio significativo respecto a los enfoques anteriores en detección de objetos. Este método ha demostrado liderazgo en resultados en

tiempo real, superando a otros algoritmos. A diferencia de técnicas como Faster RCNN, que emplean la Red de Propuestas de Región para identificar zonas de interés y luego realizar reconocimiento por separado, YOLO efectúa todas sus predicciones mediante una única capa completamente conectada. La eficiencia de YOLO radica en su capacidad para realizar todas las predicciones en una sola iteración, a diferencia de métodos que requieren múltiples iteraciones con Redes de Propuestas de Región. Este enfoque singular contribuye a su destacado rendimiento en aplicaciones de detección de objetos en tiempo real (Kundu, 2023).

Figura 4

Estructura del modelo YOLO.



Nota. Adaptado de (Datacamp, 2022)

Primero, se ajusta la imagen de entrada a un tamaño de 448x448 antes de pasar por la red convolucional. Posteriormente, se realiza una convolución de 1x1 para reducir la cantidad de canales, seguida de otra convolución de 3x3 que genera una salida con forma cúbica. La función de activación utilizada es principalmente ReLU, excepto en la

capa final, donde se emplea una función de activación lineal. Se aplican técnicas adicionales, como la normalización por lotes y la exclusión (dropout), con el propósito de regularizar el modelo y prevenir el sobreajuste (Datacamp, 2022).

De una manera más detallada, el funcionamiento de la arquitectura YOLO funciona así:

- Redimensionamiento de la imagen: La imagen de entrada se ajusta a un tamaño específico (448x448) antes de ingresar a la red. Esto puede ser importante para estandarizar el tamaño de entrada y facilitar el procesamiento.
- Convoluciones 1x1 y 3x3: Las convoluciones de 1x1 se utilizan para reducir la dimensionalidad, disminuyendo el número de canales. A esto le sigue una convolución de 3x3 que genera una salida con forma cúbica. Estas operaciones son comunes en redes neuronales convolucionales para extraer características de la imagen.
- Función de activación ReLU: ReLU (Rectified Linear Unit) se utiliza como función de activación en la mayoría de las capas, excepto en la capa final. ReLU introduce no linealidades a la red y ayuda en la convergencia del modelo.
- Función de activación lineal en la capa final: En la capa final, se opta por una función de activación lineal. Esto podría deberse a los requisitos específicos de la tarea a realizar, como la regresión.
- Regularización con normalización por lotes y dropout: La normalización por lotes ayuda a estabilizar y acelerar el entrenamiento al normalizar la entrada de cada capa. El dropout, por otro lado, desactiva aleatoriamente algunas neuronas durante el entrenamiento, lo que previene el sobreajuste al introducir variabilidad y redundancia en la red. Estas técnicas contribuyen a un modelo más generalizable y robusto.

CAPÍTULO III

MATERIALES Y MÉTODOS

En este capítulo, se abordan los temas implicados para el diseño y construcción del sistema de detección de presencia, como la arquitectura tanto en software como en hardware, el entrenamiento del modelo de visión artificial, la programación del código para poder implementar el sistema propuesto.

3.1. Diseño del sistema

En esta fase, se definen los elementos fundamentales del sistema, siguiendo la metodología en cascada que se propuso para el desarrollo de este proyecto.

3.1.1. Requerimientos

En esta sección, se identifican las especificaciones del sistema, dividiéndolas en tres categorías: requerimientos de stakeholders, requerimientos de sistema y requerimientos de arquitectura. Estos detalles se encuentran especificados en la Tabla 7.

Tabla 7

Nomenclatura para los requerimientos

NOMENCLATURA	
Requerimientos	**Abreviatura**
Stakeholders	RSSt
Funcionamiento del sistema	RFSt
Arquitectura	RARa

Nota. La tabla indica la nomenclatura a utilizar para cada uno de los requerimientos.

3.1.1.1. Determinación de Stakeholders

Según (Asana, 2023), los stakeholders del proyecto son partes interesadas que pueden influir o ser afectadas por su desarrollo. Desde colaboradores hasta ejecutivos de alto nivel, estos participantes son importantes debido a su conexión directa con el proyecto. Incluso aquellos no involucrados en tareas diarias pueden experimentar impactos derivados de los resultados. Este enfoque reconoce la importancia de los stakeholders, independientemente de su grado de participación en las actividades operativas. En la Tabla 8 se detallan cada uno de los Stakeholders del proyecto.

Tabla 8

Stakeholders del proyecto

Stakeholders	
Ing. Edwar Vásquez	Departamento de Seguridad y Gestión de Riesgo
MSc. Edgar Maya	Director del proyecto
MSc. Edgar Jaramillo	Asesor del proyecto
Sr. Juan José Vásquez	Desarrollador del proyecto

Nota. La tabla indica cada uno de los Stakeholders del proyecto.

3.1.1.2. Requerimientos de Stakeholders

En la Tabla 9, se presentan cada uno de los requerimientos de Stakeholders en base al ANEXO A, el cuál es una entrevista realizada al Ingeniero Edwar Vásquez, director del Departamento de Seguridad y Gestión de Riesgos.

Tabla 9

Requerimientos de Stakeholders

REQUERIMIENTOS DE STAKEHOLDERS (RSSt)			
N°	**Requerimientos**	**Prioridad**	
	Requerimientos del sistema	**Alta**	**Media**
RSSt 1	Sistema de bajo consumo energético		X
RSSt 2	Capturar imágenes en buena calidad	X	
RSSt 3	Sistema automatizado	X	
RSSt 4	Enviar alerta en poco tiempo	X	
RSSt 5	Operación constante	X	
	Requerimientos del Usuario		
RSSt 6	Sistema compacto		X
RSSt 7	Notificación de alerta	X	
RSSt 8	Sistema de uso intuitivo para el usuario	X	
RSSt 9	Visualización de los datos en una aplicación	X	

Nota. Requerimientos obtenidos de la entrevista a Ingeniero Edwar Vásquez.

3.1.1.3 Requerimientos de funcionamiento del sistema.

En la Tabla 10 se encuentran especificados los requerimientos planteados para el óptimo funcionamiento del sistema. Estos requisitos, tienen como objetivo abordar de manera integral las distintas necesidades del usuario. Cabe destacar que dichos requerimientos se dividen en categorías específicas, tales como: requerimientos de uso, interfaces y desempeño, proporcionando así un enfoque detallado y completo para la implementación del sistema.

Tabla 10

Requerimientos de funcionamiento del sistema

REQUERIMIENTOS DE FUNCIONAMIENTO (RFSt)			
N°	**Requerimientos**	**Prioridad**	
	Requerimientos de Uso	**Alta**	**Media**
RFSt 1	El sistema debe tener una fuente de alimentación		X
RFSt 2	Sistema de uso intuitivo para el usuario	X	
RFSt 3	Toma datos de forma rápida	X	
RFSt 4	Inicio del sistema desde la aplicación y cuando es requerido	X	
	Requerimientos de interfaces		
RFSt 5	Puerto de conexión para cámara	X	
RFSt 6	Puerto Ethernet o interfaz inalámbrica para conexión a internet	X	
	Requerimientos de Desempeño		
RFSt 7	Visualiza la imagen adquirida y analizada junto con la alerta	X	
RFSt 8	Escalabilidad del sistema	X	
RFSt 9	Alta disponibilidad	X	
RFSt 10	El sistema envía la alerta en menos de 1 minuto	X	
RFSt 11	El sistema es capaz de detectar personas y contarlas	X	

Requerimientos Físicos			
RFSt 12	La cámara es capaz de capturar imágenes de forma nítida	X	
RFSt 13	El sistema debe ser compacto		X
RFSt 14	El sistema cuenta con una fuente de alimentación	X	
RFSt 15	El sistema cuenta con una carcasa de protección	X	

Nota. La tabla muestra cada uno de los requerimientos funcionales del sistema.

3.1.1.4 Requerimientos de la arquitectura

En la Tabla 11 se detallan cada uno de los requerimientos de arquitectura, los cuales describen las características esenciales del diseño de software y hardware que son necesarias para lograr que el sistema cumpla con los objetivos del proyecto.

Tabla 11

Requerimientos de arquitectura

REQUERIMIENTOS DE ARQUITECTURA (RARa)			
N°	**Requerimientos**	**Prioridad**	
	Requerimientos de Diseño	**Alta**	**Media**
RARa 1	La cámara se ubica en un punto estratégico	X	
RARa 2	El sistema tiene escalabilidad	X	
RARa 3	Sistema de bajo costo		X
RARa 4	El sistema tiene una fuente de alimentación propia		X
RARa 5	El sistema tiene conexión a internet	X	

Requerimientos Lógicos		
RARa 6	El sistema cuenta el número de personas detectadas	X
RARa 7	El sistema detecta personas de manera precisa	X
RARa 8	Sistema entrenado para trabajar en distintos escenarios	X
RARa 9	Reestablece conexión con la API para el envío de datos	X
Requerimientos de Software		
RARa 10	Sistema operativo de software libre	X
RARa 11	Lenguaje de programación de código abierto	X
RARa 12	Librerías compatibles con el sistema operativo y el lenguaje de programación	X
Requerimientos de Hardware		
RARa 10	Sistema embebido con puerto CSI para una cámara	X
RARa 11	Cámara compatible con el Sistema embebido	X
RARa 12	La fuente de energía alimenta todo el sistema	X
RARa 13	Sistema embebido con procesador y memoria de alta capacidad	X

RARa 14	El sistema cuenta con un ventilador para refrigeración	X

Nota. La tabla describe todos los requerimientos de arquitectura que debe tener el sistema de detección de presencia de personas.

3.2. Diseño

3.2.1. Descripción general del sistema

El proyecto por desarrollarse propone un sistema de detección de presencia de personas que, mediante visión artificial, identifique si hay o no personas dentro de un aula en la Universidad Técnica del Norte. El sistema notificará mediante una aplicación móvil al encargado del personal de seguridad para que, en base al resultado obtenido, se realicen las acciones correspondientes al plan de emergencias de la universidad.

El sistema cuenta con una placa que tendrá conectada una cámara para poder tomar las fotos del lugar en el que se encuentre instalada. Estas imágenes serán analizadas por el sistema que utiliza técnicas de aprendizaje profundo, con un dataset de imágenes etiquetado y entrenado con un modelo de visión artificial. El sistema determinará si en la imagen tomada existe la presencia de una persona o no. En caso de que se detecte la presencia, se contará cuántas personas hay. El sistema enviará un mensaje de alerta por medio de una aplicación móvil. Este mensaje contendrá la imagen tomada y analizada, junto con el resultado obtenido por el sistema indicando si hay una persona adentro y cuántas hay en caso de que exista presencia de una persona dentro del aula.

El sistema estará conectado a la aplicación móvil Telegram mediante una API KEY generada por el bot configurado dentro de la aplicación. En este bot se creará el comando que luego será programado dentro del sistema. Este comando será utilizado por el personal de seguridad para iniciar el sistema y comenzar a detectar la presencia de personas.

3.2.2. Arquitectura del sistema

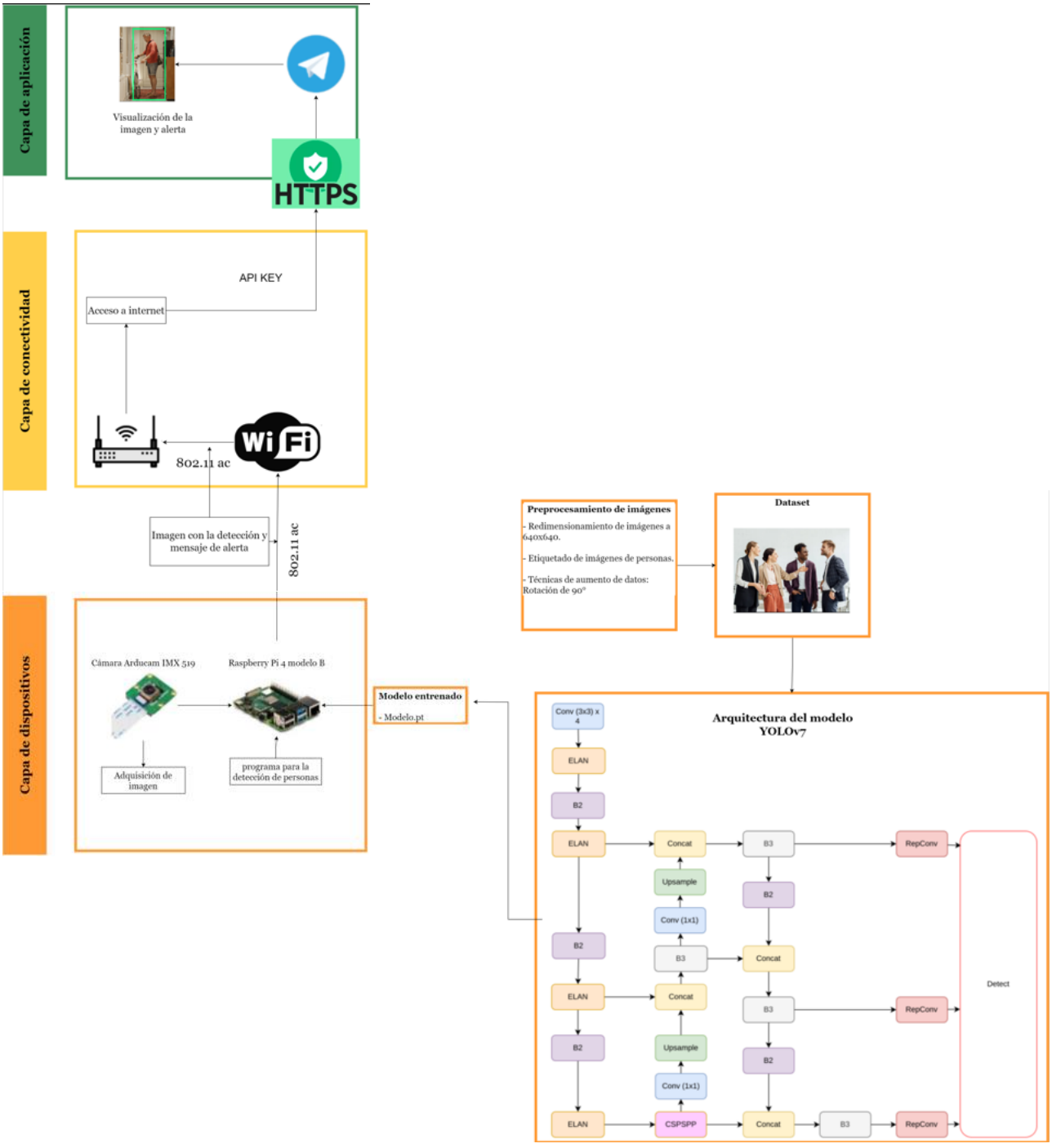
A continuación, se presenta el diagrama de bloques del sistema de detección de presencia. Se toma como referencia una arquitectura de 3 capas IoT, que se detallan a continuación con el fin de utilizarlas para el diseño del sistema.

En cuanto a la capa de dispositivos, se considera la cámara la cual está encargada de adquirir la imagen , que va conectada al sistema embebido. En este sistema, se encuentra el modelo entrenado previamente con un dataset de imágenes de personas etiquetado con una clase, así como también el programa donde se realiza la detección de personas y la generación de la alerta.

En la capa de conectividad o gateway, se usa el protocolo de comunicación inalámbrica IEEE 802.11 ac para conectarse a un punto de acceso que estará ubicado dentro del aula donde se encuentre el sistema. Con este punto de acceso, el sistema tendrá conexión a internet. Mediante una API KEY y usando el protocolo HTTPS, el sistema se conectará a la aplicación móvil Telegram. En esta aplicación, se visualizarán los mensajes de alerta que enviará el sistema en caso de detectar la presencia de una persona. Esta capa, que pertenece a la capa aplicación, es la última de la arquitectura. En la Figura 5 se aprecia la arquitectura del sistema de detección de presencia.

Figura 5

Arquitectura del sistema



3.2.3. Sistema embebido

En este caso se usa el sistema embebido Raspberry Pi 4 modelo B, debido a que es una placa SBC, con una buena potencia de procesamiento y memoria que permite ejecutar diferentes aplicaciones de visión artificial desde las más sencillas hasta las más complejas y así, manejar conjuntos de datos robustos. Su tamaño compacto, portabilidad y bajo costo son otros factores tomados en cuenta para el uso de este sistema embebido. En la Tabla 1 del capítulo 2 se encuentra de manera detallada las especificaciones técnicas del sistema embebido.

3.2.4. Cámara

Se utiliza la cámara Arducam IMX519 por su resolución de 16 Megapíxeles y su buen rendimiento en condiciones donde hay poca luz, ya que su sensor CMOS Sony IMX519 tiene un nivel bastante alto de sensibilidad, y permite su uso en aplicaciones que requieren de visión precisa, como lo son el reconocimiento de objetos, o en este caso personas. Su compatibilidad con Raspberry Pi 4 modelo B hace un gran complemento a la hora de construir el sistema de detección de presencia de personas. En la Tabla 3 del capítulo 2 se detallan sus especificaciones técnicas.

3.2.5. Sistema Operativo

El sistema operativo que se escogió es Raspberry Pi OS, la principal razón es debido a su gran optimización y compatibilidad ya que está optimizado específicamente para dispositivos Raspberry Pi, lo que puede significar que aprovechará más los recursos de este sistema embebido en cuanto a hardware, tiene muy buen soporte comunitario, y

es bastante intuitivo, sobre todo para los usuarios nuevos que están comenzando a usar este tipo de sistemas embebidos.

3.2.6. Lenguaje de programación

Para este proyecto, se ha seleccionado Python como el lenguaje de programación principal debido a su amplia variedad de bibliotecas especializadas en visión artificial, como OpenCV y Pytorch. Python se destaca por su sintaxis clara y accesible, facilitando tanto el aprendizaje para principiantes como la rápida implementación de algoritmos. Además, la compatibilidad de Python con las librerías específicas para sistemas embebidos, como la Raspberry Pi, garantiza una integración eficiente con los componentes de hardware, como la cámara oficial de Raspberry Pi.

3.2.7. Modelo YOLOv5

YOLOv5 pertenece a la familia de modelos de visión por computadora conocida como You Only Look Once (YOLO). Este modelo se emplea habitualmente para la detección de objetos. YOLOv5 está disponible en cuatro versiones distintas: pequeña (s), mediana (m), grande (l) y extragrande (x). Cada una de estas versiones proporciona niveles crecientes de precisión. Asimismo, cada variante demanda un tiempo de entrenamiento diferente (Solawetz, 2020). Es un modelo muy popular y de alto rendimiento en el campo de detección de objetos, es considerado como la tecnología de punta en detecciones en tiempo real (FPS). YOLO-v5 es la quinta generación de los detectores de una sola etapa. YOLO-v5 está implementado en Pytorch (Montenegro et al., 2022).

YOLOv5 es un detector de objetos que funciona mediante la extracción de características de imágenes. Estas características se utilizan luego para identificar los objetos en la imagen y predecir su clase.

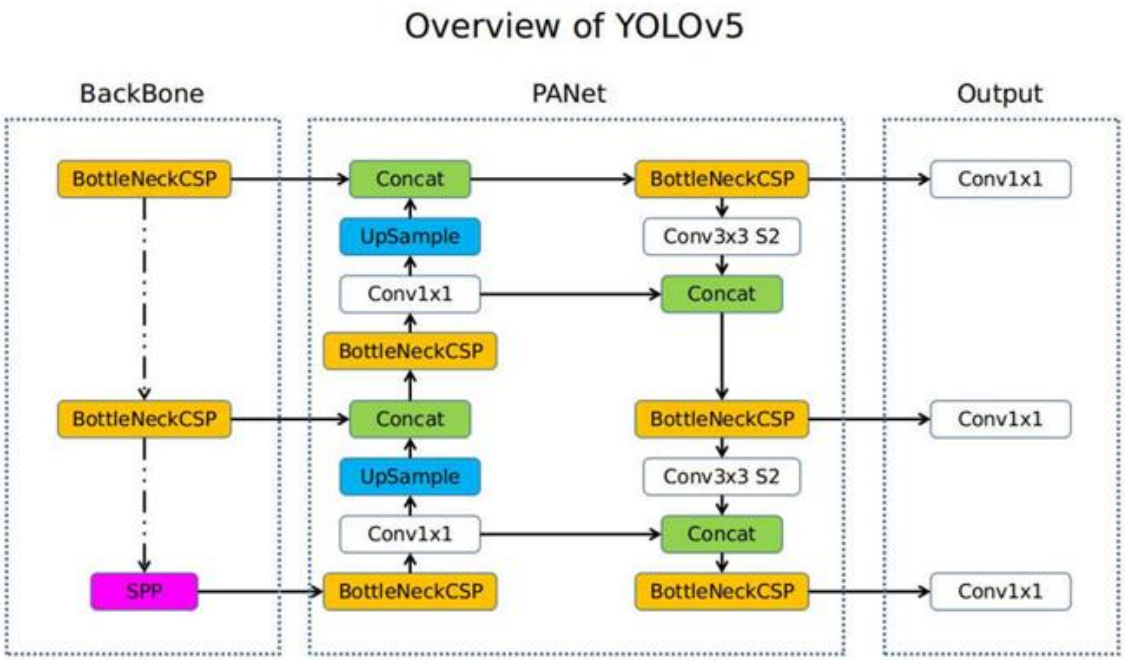
Según (Solawetz, 2020), el modelo YOLOv5 consta de 3 partes principales:

- **Backbone:** La backbone es la sección de la red responsable de obtener las características de la imagen. En este contexto, la backbone emplea una arquitectura de redes convolucionales profundas (CNN) conocida como CSPDarknet53.
- **Cuello:** Un conjunto de capas destinadas a integrar y combinar características de la imagen antes de ser utilizadas en el proceso de predicción.
- **Cabeza:** Esta sección de la red es responsable de producir las predicciones. En este escenario, la salida genera tres mapas de predicción, cada uno representando una escala diferente de la imagen.

La arquitectura de YOLOv5 se puede observar de manera más detallada en la siguiente figura.

Figura 6

Arquitectura del modelo YOLOv5



Nota. Adaptado de (Gardner, 2021)

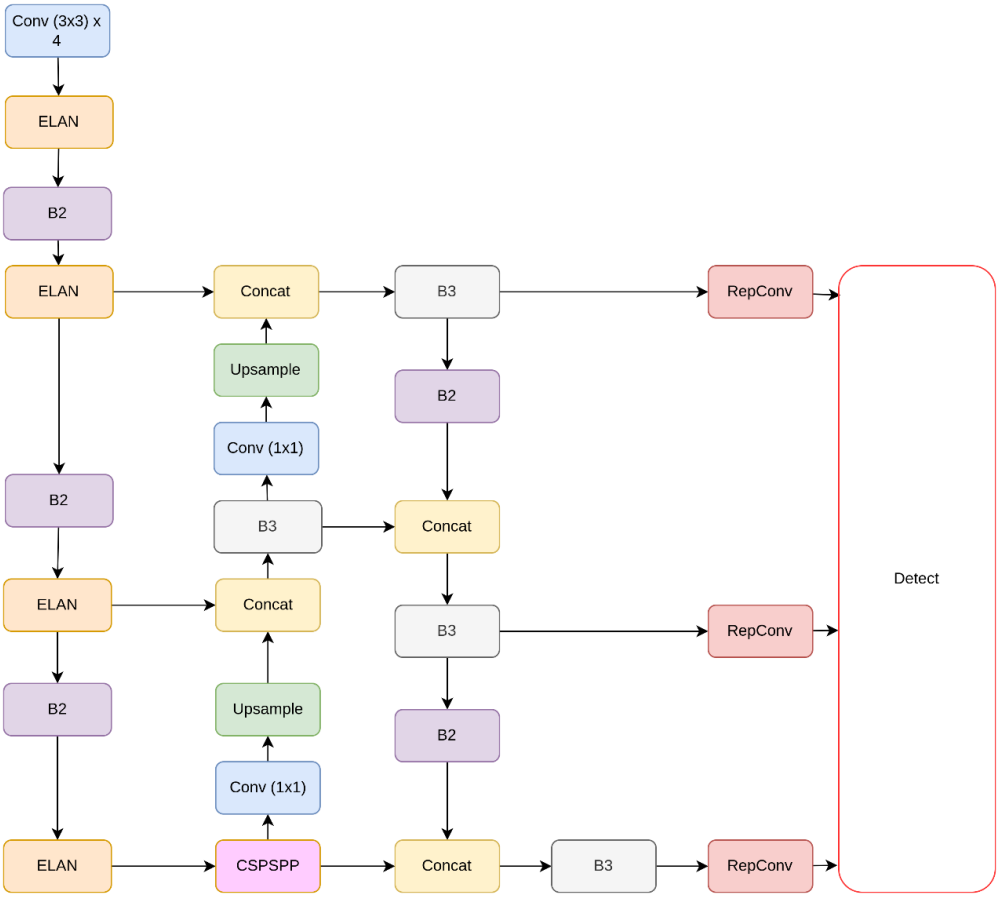
Existen múltiples maneras de combinar diversas arquitecturas en cada componente clave. Las aportaciones de YOLOv5 se centran en incorporar avances provenientes de otras áreas de visión por computadora y en mostrar que, en conjunto, mejoran la capacidad de detección de objetos de YOLO.

3.2.8. Modelo YOLOv7

YOLOv7 se presenta como un detector de objetos en tiempo real de vanguardia, superando a todos los detectores de objetos conocidos tanto en velocidad como en precisión, abarcando un rango desde 5 FPS hasta 160 FPS. Destaca por alcanzar la máxima precisión (56.8% AP) entre los detectores de objetos en tiempo real conocidos, especialmente logrando 30 FPS o más en la GPU V100. Además, supera en rendimiento a otros detectores de objetos como YOLOR, YOLOX, Scaled-YOLOv4, YOLOv5, entre otros, tanto en velocidad como en precisión. Es relevante señalar que este modelo se entrena desde cero exclusivamente con el conjunto de datos MS COCO, prescindiendo de otros conjuntos de datos o pesos preentrenados (Ultralytics, 2022).

Figura 7

Arquitectura modelo YOLOv7



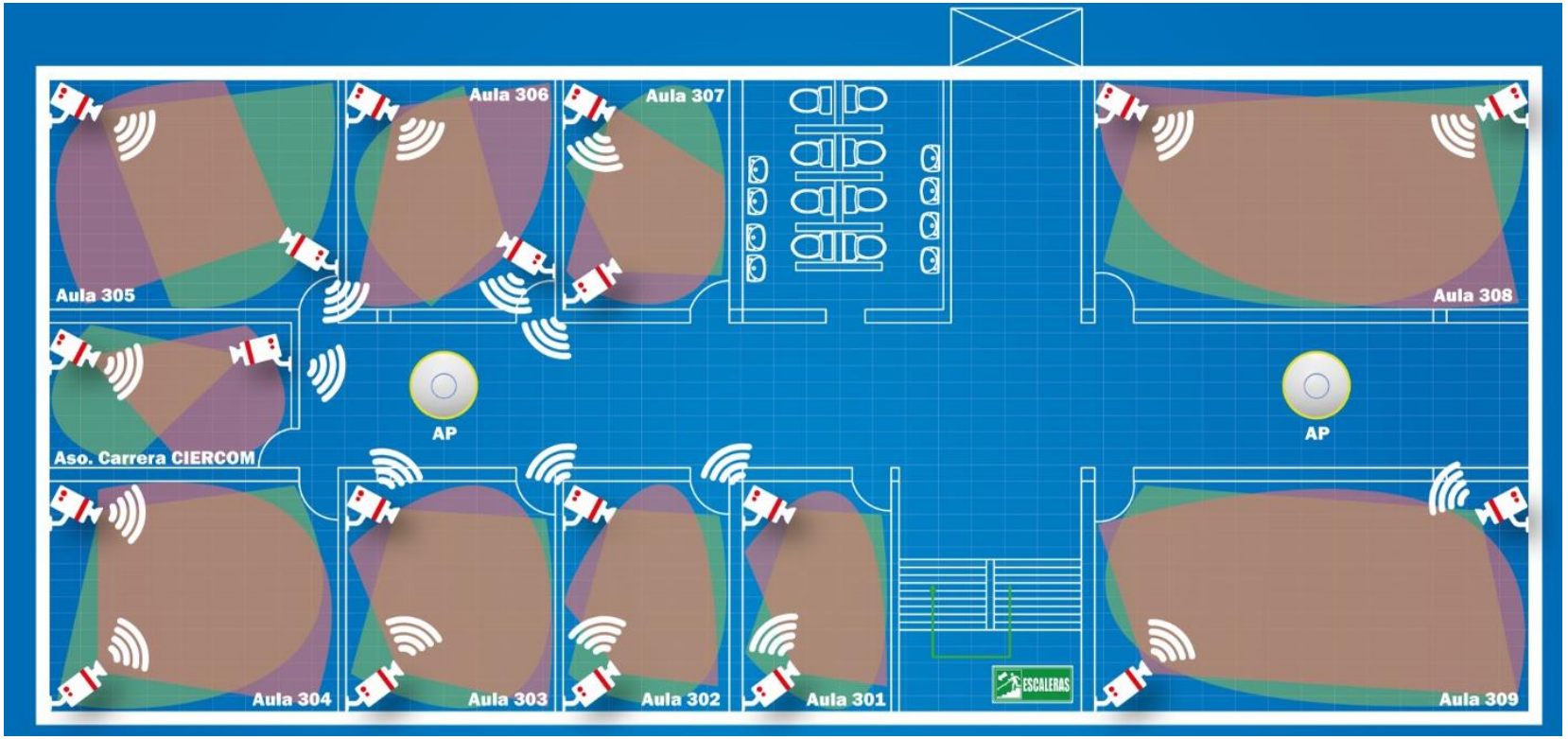
Nota. *Adaptado de* (Weerasinghe, 2022)

3.2.8. Planos de implementación del sistema

En la siguiente figura se muestran los planos de implementación del sistema de detección de presencia de personas, dentro del cuarto piso de la Facultad de Ingeniería y Ciencias Aplicadas (FICA).

Figura 8

Planos de la implementación del sistema 4to piso FICA



Para poder comprender cada uno de los elementos que se encuentran dentro del plano se debe tomar en cuenta la siguiente nomenclatura:

Figura 9

Nomenclatura del plano de implementación

Para poder implementar el sistema lo primero que se realizó fue el cálculo del campo de visión horizontal que tiene la cámara, ya que es importante saber que área es capaz de cubrir la cámara dentro del aula. Para ello se ha utilizado la siguiente fórmula que permite realizar este cálculo:

$$FOV_{horizontal} = 2 \times \arctan\left(\frac{Tamaño\ del\ sensor}{2 \times Distancia\ focal}\right)$$

En este caso el sensor que se utiliza es el Sony IMX519 el cuál tiene un tamaño de 1/2.6 pulgadas. Según el datasheet de la cámara, la lente tiene una distancia focal de 4.28 mm.

Convirtiendo el valor del ancho del sensor en pulgadas a milímetros, se tiene un valor aproximado de 6.153 mm.

Reemplazando en la fórmula se tiene:

$$FOV_{horizontal} = 2 \times \arctan\left(\frac{6.153\ mm}{2\ \times 4.28\ mm}\right)$$

$$FOV_{horizontal} = 2 \times \arctan\left(0.718\right)$$

$$FOV_{horizontal} = 80°$$

En base a este resultado se diseña como irán implementadas cada una de las cámaras tomando en cuenta este campo de visión, por lo cuál se necesitan de dos cámaras como mínimo para cubrir casi en su totalidad el aula.

En la siguiente tabla se detalla de mejor manera esta parte:

Tabla 12

Cámaras necesarias para la implementación

Aula	N° de cámaras	Área por cubrir
301	2	31,20 m²
302	2	33,55 m²
303	2	40,26 m²
304	2	48,64 m²
305	2	48,64 m²
306	2	40,26 m²
307	2	33,55 m²
308	2	74,481 m²
309	2	74,481 m²
ASO-CITEL	2	21,75 m²

Otro valor para tomar en cuenta en esta etapa de diseño es calcular el valor estimado de la distancia hasta la que la cámara podrá detectar de manera clara los objetos al tomar las fotos, lo que infiere en el proceso de detección del sistema con visión artificial.

Para ello, se usa la siguiente expresión:

$$D = \frac{Resolución\ del\ sensor\ \times Tamaño\ del\ objeto}{Tamaño\ del\ objeto\ en\ la\ imagen}$$

Cabe recalcar que todo esto es un calculo estimado ya que no siempre el tamaño del objeto será el mismo en todos los casos. Para este ejemplo se toma un tamaño del objeto de 1 metro de ancho, que, dentro de la imagen, el objeto ocupa aproximadamente 1000 píxeles de ancho. Reemplazando se tiene lo siguiente:

$$D = \frac{4608Px\ \times 1m}{1000\ Px}$$

$$D = 4{,}608\ m$$

Para la parte inalámbrica del sistema se tiene establecido la asignación de canales y en que banda van a trabajar cada una de las cámaras que se van a conectar al punto de acceso para enviar las alertas.

Tabla 13

Tabla de diseño de la parte inalámbrica del sistema

Aula	Cámara	Canal	Banda	Punto de acceso
301	1	1	2.4 GHz	AP pasillo
	2	6	2.4 GHz	AP pasillo
302	1	11	2.4 GHz	AP pasillo
	2	1	2.4 GHz	AP pasillo

303	1	6	2.4 GHz	AP pasillo
	2	11	2.4 GHz	AP pasillo
304	1	12	2.4 GHz	AP pasillo
	2	13	2.4 GHz	AP pasillo
305	1	1	2.4 GHz	AP pasillo
	2	6	2.4 GHz	AP pasillo
306	1	11	2.4 GHz	AP pasillo
	2	12	2.4 GHz	AP pasillo
307	1	13	2.4 GHz	AP pasillo
	2	1	2.4 GHz	AP pasillo
308	1	6	2.4 GHz	AP Lab Fibra
	2	11	2.4 GHz	AP Lab Fibra
309	1	12	2.4 GHz	AP Lab Fibra
	2	13	2.4 GHz	AP Lab Fibra
ASO-CITEL	1	1	2.4 GHz	AP pasillo
	2	6	2.4 GHz	AP pasillo

Cada uno de los canales asignados para cada dispositivo, se lo realiza con el propósito de minimizar la interferencia, usando canales no superpuestos.

3.2.9. Diagrama de flujo

3.2.9.1. Diagrama de flujo del proceso de generación del dataset de imágenes

El primer paso en el desarrollo de un sistema de visión artificial implica la creación del conjunto de datos de imágenes. Es crucial contar con un conjunto inicial de

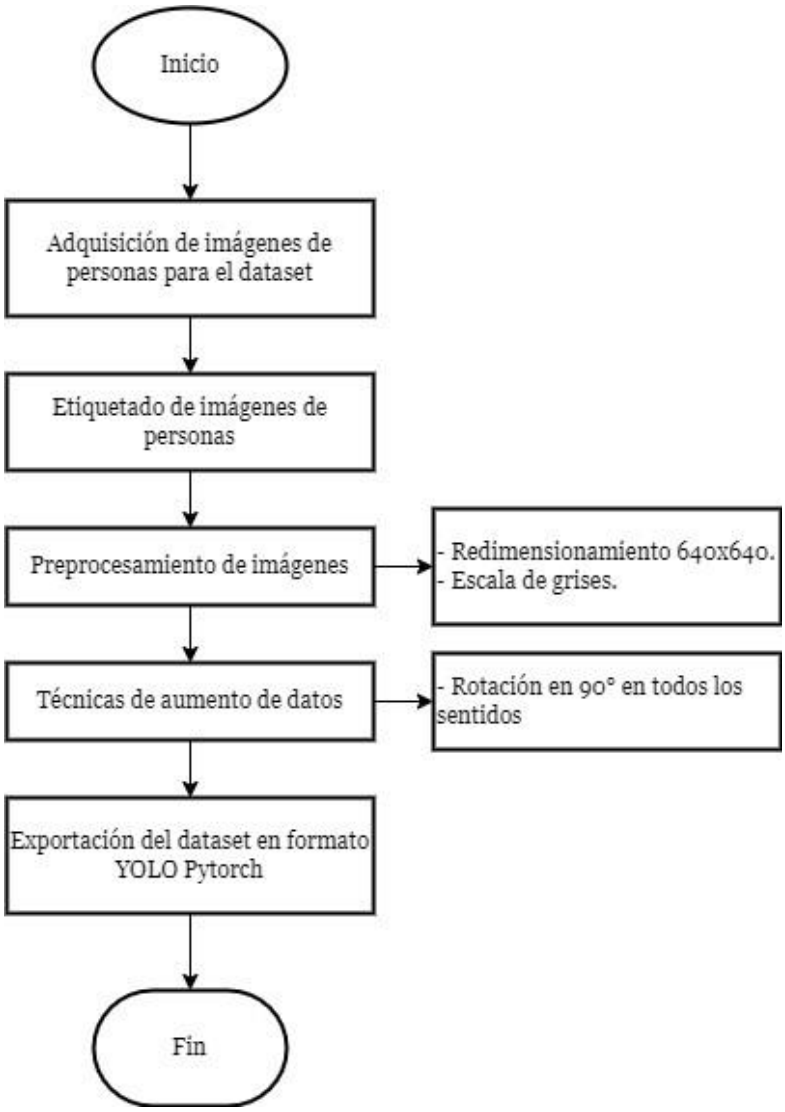
alta calidad, especialmente en este caso, que se centra en imágenes de personas. Una vez obtenido este conjunto, el siguiente paso es etiquetar cada imagen, ya que estas etiquetas forman la base fundamental para el aprendizaje del modelo.

Después de etiquetar las imágenes, se procede al preprocesamiento, que incluye la aplicación de técnicas de aumento de datos. Estas técnicas son esenciales para generar un conjunto de datos más robusto al crear variaciones en las imágenes. Finalmente, el dataset se exporta en formato YOLO PyTorch, lo que facilita su uso en el proceso de entrenamiento.

En la Figura 10 se presenta visualmente el proceso a seguir para la generación del dataset

Figura 10

Diagrama de flujo del proceso para generar el dataset de imágenes de personas



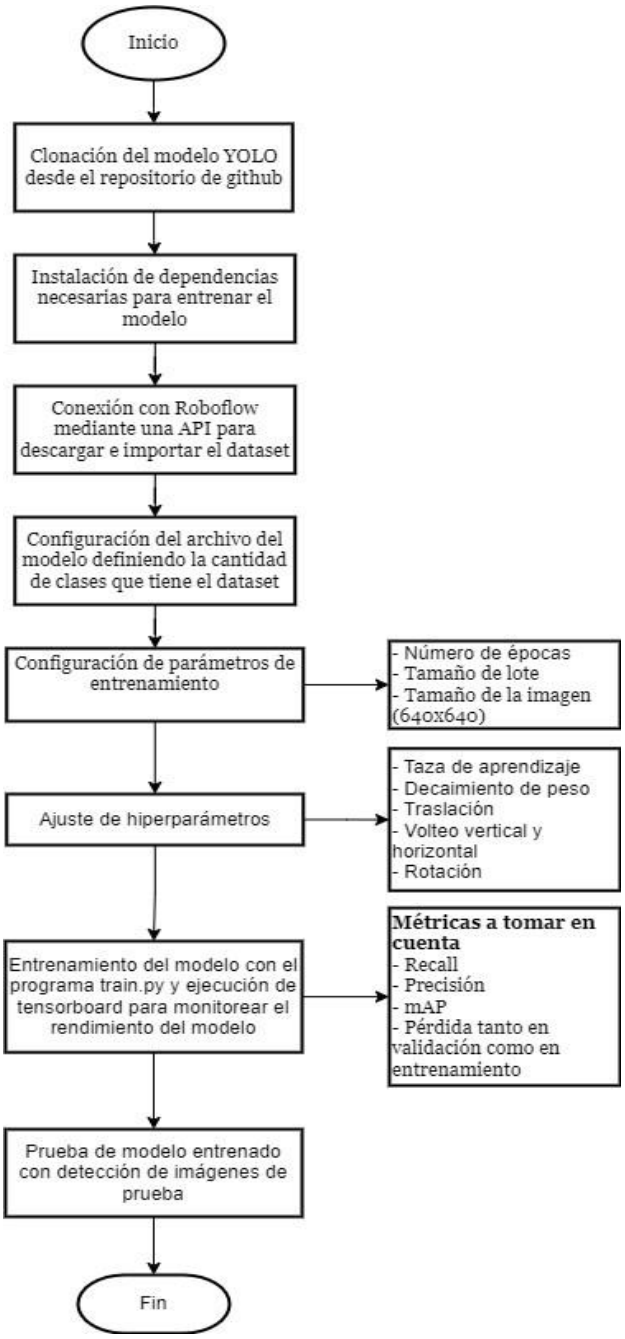
Nota. Autoría propia.

3.2.9.2. Diagrama de flujo del proceso de entrenamiento del modelo con YOLOv5 y YOLOv7

Para llevar a cabo el entrenamiento del modelo utilizando YOLO, el primer paso consiste en acceder a un cuaderno en Google Colab. En este cuaderno, el procedimiento inicial implica clonar el modelo desde su repositorio en GitHub, seguido de la instalación de todas las dependencias necesarias. Una vez completado este proceso, se procede a ingresar a la plataforma Roboflow, donde se exporta el modelo utilizando una API KEY. Esta clave posibilita la descarga e importación del conjunto de datos de imágenes creado previamente, esencial para el entrenamiento del modelo. Para obtener una descripción más detallada del proceso de entrenamiento, se puede consultar la Figura 11.

Figura 11

Diagrama de flujo del proceso de entrenamiento del modelo con YOLO



Nota. Autoría propia

3.2.9.3. Diagrama de flujo del programa de sistema de detección de personas

Dentro del programa, como primer paso se importan cada una de las librerías con las que se va a trabajar, en la Tabla 14 se encuentran especificadas cada una de ellas. Luego se carga el modelo entrenado con YOLO en formato Pytorch y se importa la clave de la API de Telegram. Si el programa recepta que se envió el comando '/foto' desde el chat del bot, ejecuta la cámara y toma una foto en una resolución de 4656x3496 y la guarda en un directorio. Se lee la imagen, se la preprocesa redimensionando la imagen a una entrada de 640x640 y se la normaliza. Se ejecuta la inferencia del modelo para realizar predicciones aplicando NMS para que sea más precisa la detección. Se guarda la imagen con la detección realizada y se la envía al chat de telegram. Debido a que, por la inactividad del dispositivo, puede caerse la conexión con el bot, el programa tiene un máximo de 3 intentos para restablecer la conexión y enviar los mensajes de alerta ya que al tener problemas de conexión los mensajes de alerta no se envían y únicamente se envía la imagen. Este proceso se evidencia de mejor manera en el diagrama de flujo de la Figura 12.

Tabla 14

Librerías utilizadas para el programa de sistema de detección de personas

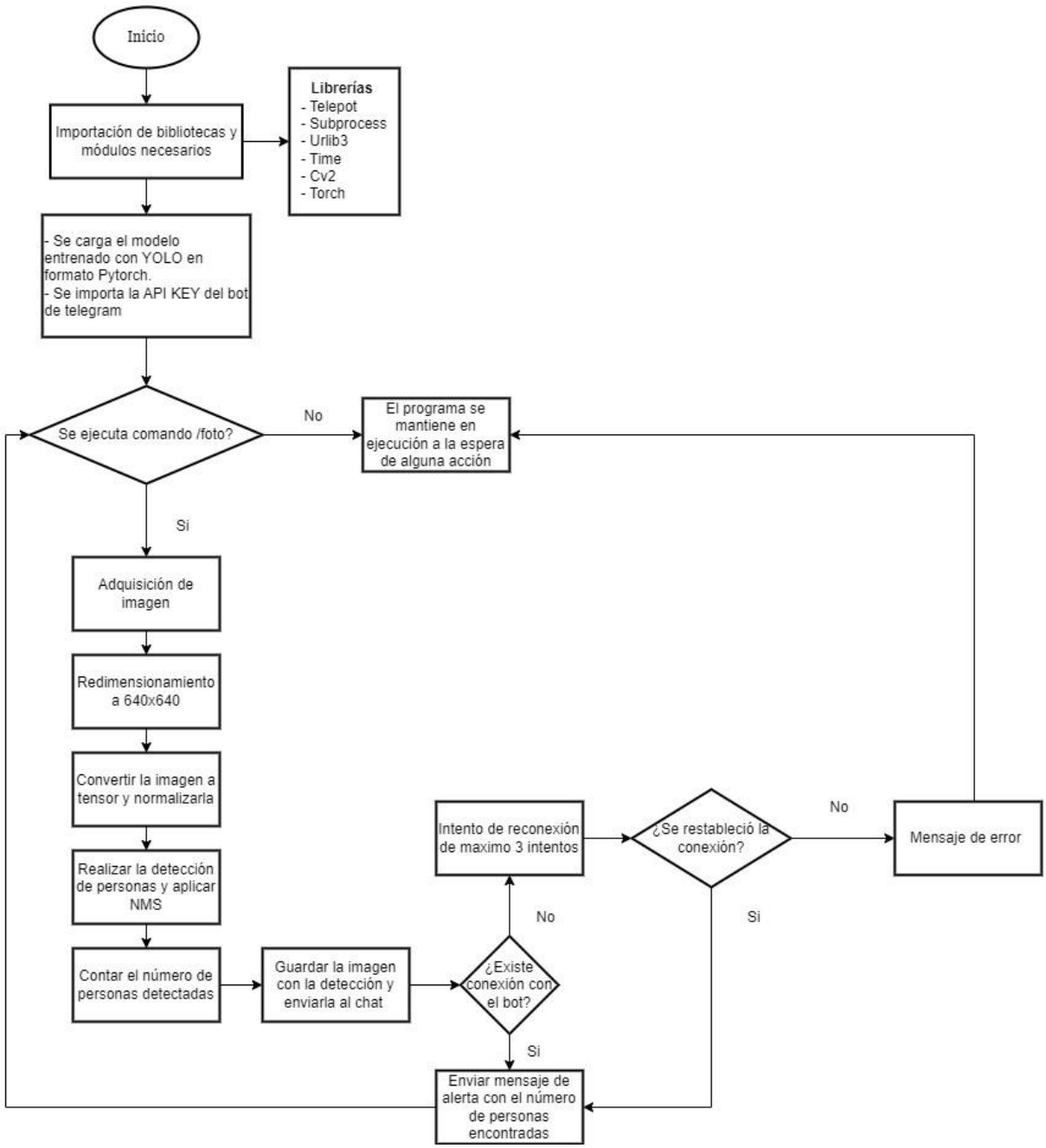
Librería	Descripción
Telepot	Interactuar con la API de Telegram y gestionar bots, facilitando el envío de mensajes y contenido multimedia.
Subprocess	Ejecutar comandos del sistema operativo desde Python, permitiendo la interacción con procesos externos.

Urllib3	Realizar solicitudes HTTP de manera eficiente, gestionando conexiones y manejando URLs.
Time	Manipular funciones relacionadas con el tiempo, como medición de tiempo y pausas de ejecución.
Torch	Biblioteca de aprendizaje profundo que incluye estructuras de datos para tensores y operaciones de redes neuronales.
CV2	Es una librería de visión por computadora que proporciona herramientas para el procesamiento de imágenes y videos.

Nota. En la tabla se describen cada una de las librerías utilizadas para el programa de sistema de detección de personas. Fuente: Autoría propia

Figura 12

Diagrama de flujo del programa de sistema de detección de personas



Nota. Autoría propia

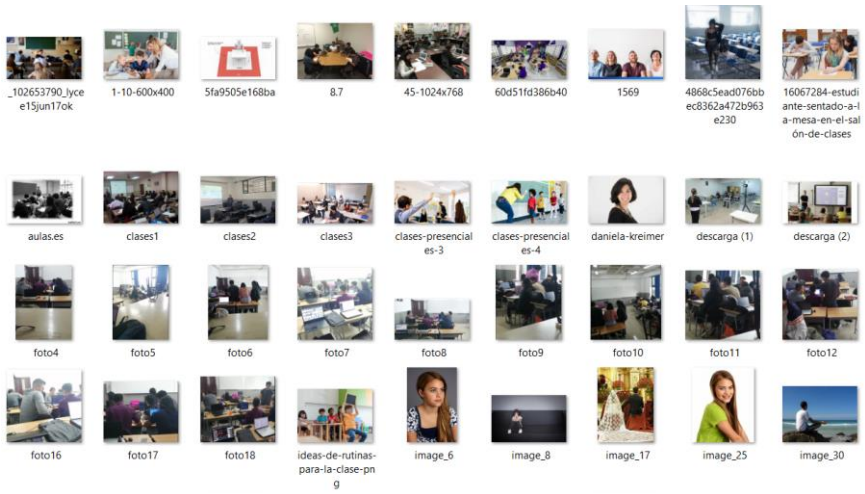
3.2.10. Generación del dataset de imágenes para entrenamiento

3.2.10.1. Dataset de imágenes de personas

El primer paso para implementar el sistema propuesto es adquirir imágenes de personas. Inicialmente, se capturaron imágenes dentro del aula donde se implementará el sistema, abarcando diversos escenarios. Dado que el dataset es de una sola clase, es esencial capturar situaciones que permitan un entrenamiento efectivo del modelo. Se incluyeron fotografías con el aula llena, con una cantidad media de personas, con pocas personas e incluso una sola, en condiciones de mucha luz (ya sea ambiental o con la luz encendida) y en escenarios con poca luz, adicional a estas imágenes se usó un dataset de imágenes de personas obtenido de la plataforma Roboflow, así como imágenes añadidas por búsqueda propia para tener un dataset bastante grande con el que se pueda entrenar el modelo. El conjunto inicialmente comienza 700 imágenes, ya que con el aumento de datos que se realiza antes de generar el dataset, tendrá un incremento.

Figura 13

Dataset de imágenes de personas



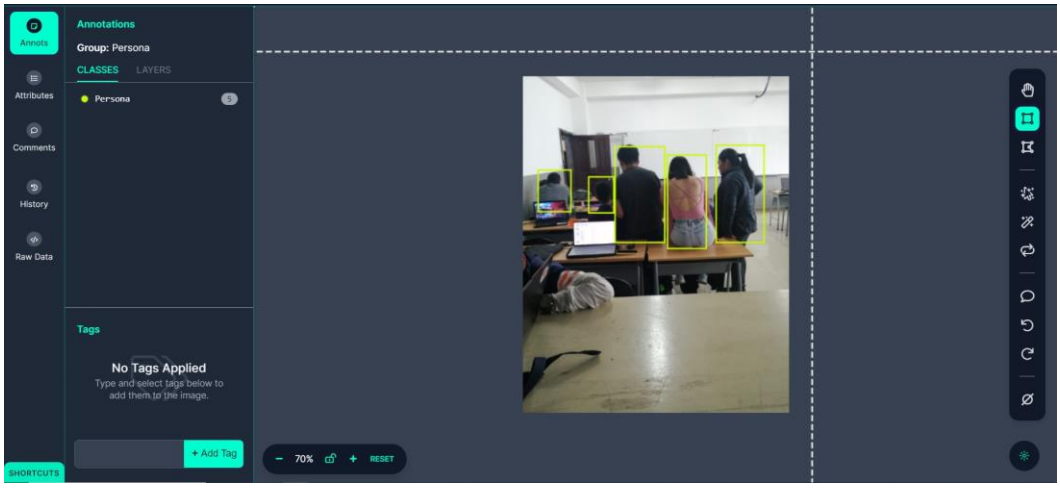
3.2.10.2. Etiquetado de imágenes

Para llevar a cabo el etiquetado de las imágenes en el dataset, se optó por utilizar la plataforma Roboflow debido a su accesibilidad, siendo gratuita y contando con una interfaz intuitiva que facilita el proceso de etiquetado. Inicialmente, se cargaron las imágenes recopiladas previamente en la plataforma, y luego se procedió a etiquetarlas una por una.

En Roboflow, se definió la clase de interés, en este caso, 'persona', y se utilizó la herramienta de etiquetado para encerrar dentro de un recuadro la región de la imagen que contiene a la persona. Este enfoque de etiquetado se ajusta a la tarea específica de detección de personas en imágenes. En la Figura 14 se puede observar el etiquetado de las imágenes en Roboflow.

Figura 14

Etiquetado de imágenes en Roboflow



Nota. Autoría propia

3.2.10.3. Preprocesamiento de imágenes

Una vez etiquetadas todas las imágenes de personas en el conjunto de datos, se avanza a la fase de establecer la distribución porcentual que tendrá el dataset. Esto es

esencial para dividir adecuadamente las imágenes entre los conjuntos de entrenamiento, validación y pruebas.

Figura 15

Porcentaje de distribución de imágenes entre entramiento, validación y pruebas.

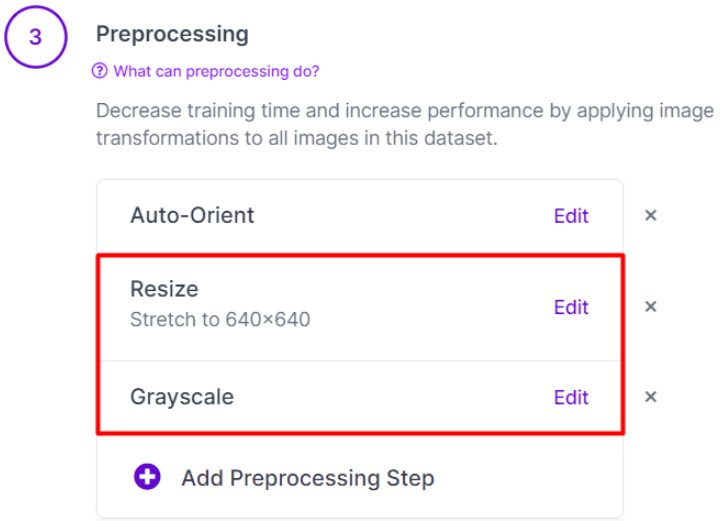
Nota. Autoría propia

Después de agregar las imágenes al conjunto de datos, se lleva a cabo el preprocesamiento correspondiente. Para facilitar este proceso de manera intuitiva, se utiliza la plataforma Roboflow.

Este conjunto de datos se ha sometido a dos técnicas de preprocesamiento. En primer lugar, se redimensionaron las imágenes a un tamaño de 640x640. Además, se aplicó la conversión a escala de grises. Estas acciones tienen como objetivo principal reducir el tiempo de procesamiento durante el entrenamiento y mejorar la precisión del modelo ante posibles cambios en la iluminación y el contraste. Se redimensiona la imagen a la resolución de 640x640 debido a que este es el tamaño de entrada base que acepta el modelo YOLOv7, aunque se pueden usar tamaños más bajos, es preferible usar el tamaño base ya que es una buena resolución que permite al modelo detectar de manera más clara los objetos cuando están más lejos.

Figura 16

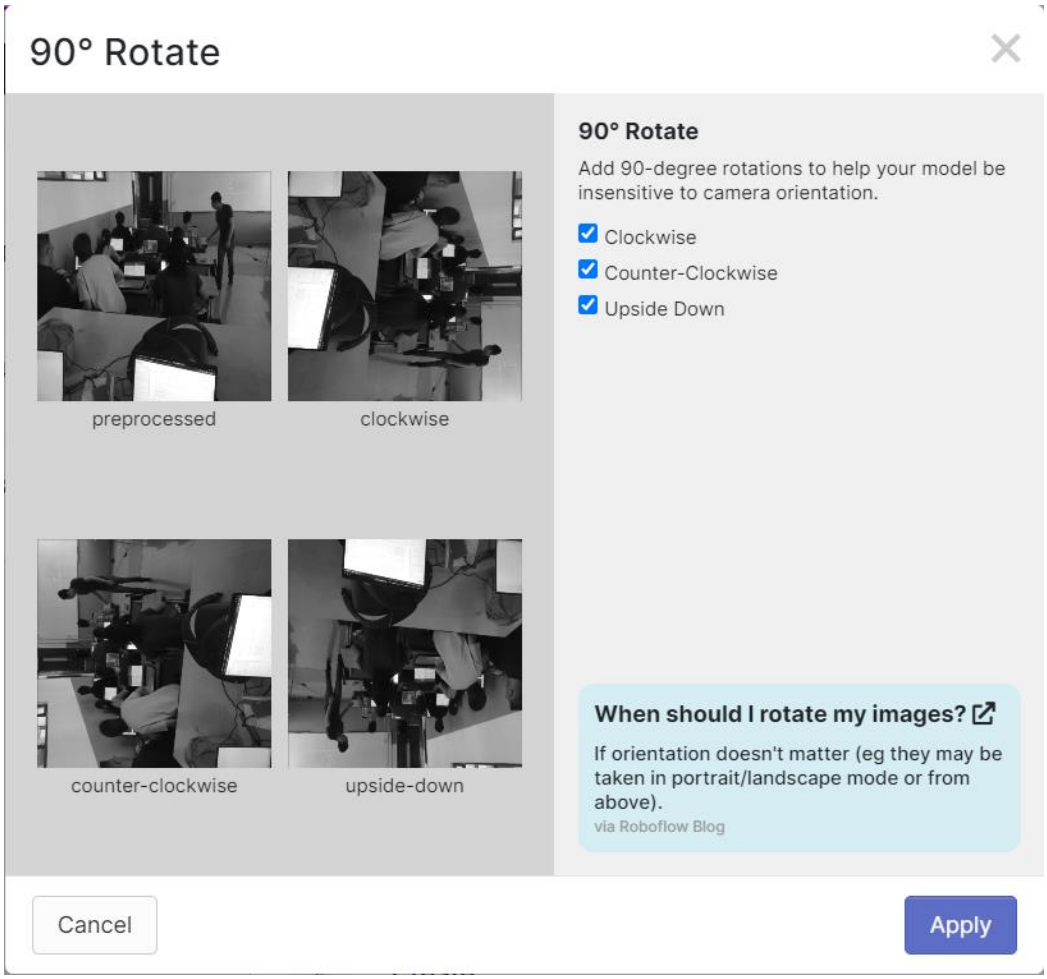
Técnicas de preprocesamiento de imágenes



Para garantizar un conjunto de datos más extenso y permitir que el modelo de entrenamiento aprenda de una variedad de escenarios, se aplican técnicas de aumento de datos, para este proyecto se utiliza la rotación de imágenes en 90° en todos los sentidos. Este enfoque busca mejorar la precisión del modelo al exponerlo a diferentes situaciones. La técnica de aumento de datos utilizadas para este dataset se puede observar en la siguiente figura.

Figura 17

Técnica de aumento de datos para las imágenes



3.2.11. Entrenamiento de la red neuronal con YOLOv5

El entrenamiento de esta versión del modelo de YOLO comienza con la clonación del repositorio oficial de ultralytics, ya que contiene cada uno de los requerimientos y dependencias necesarias para realizar el entrenamiento. Como se observa en la Figura 18.

Figura 18

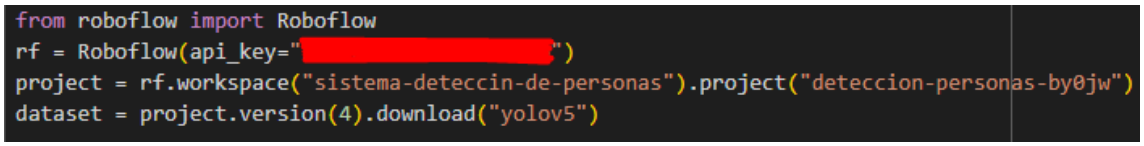
Clonación del repositorio del modelo YOLOv5

```
[ ]  #clone YOLOv5 and
     !git clone https://github.com/ultralytics/yolov5  # clone repo
     %cd yolov5
     %pip install -qr requirements.txt # install dependencies
     %pip install -q roboflow

     import torch
     import os
     from IPython.display import Image, clear_output  # to display images

     print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name if torch.cuda.is_available() else 'CPU'})")
```

Una vez realizada la instalación de los requerimientos, se debe importar el dataset generado anteriormente en la plataforma Roboflow, mediante el uso de una API KEY generada por la misma plataforma.

Figura 19

Importación del dataset generado en la plataforma Roboflow



```
from roboflow import Roboflow
rf = Roboflow(api_key="[redacted]")
project = rf.workspace("sistema-deteccin-de-personas").project("deteccion-personas-by0jw")
dataset = project.version(4).download("yolov5")
```

Antes de comenzar con el entrenamiento como tal, se proceden a configurar algunos archivos. Para YOLOv5 se realiza el entrenamiento con dos versiones, la versión ‘s’ y la versión ‘x’, con el fin de encontrar el mejor modelo para este proyecto.

En cuanto al archivo de configuración, se deben realizar cambios de los archivos de configuración de las dos versiones del modelo. Estableciendo que el número de clases es 1, ya que por defecto vienen 80. En la Figura 20 y Figura 21 se observa la configuración de los archivos yolov5s.yaml y yolov5x.yaml.

Figura 20

Configuración del archivo yolov5s.yaml

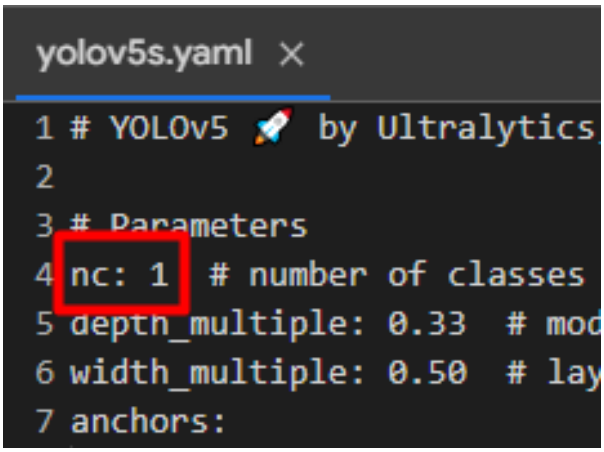
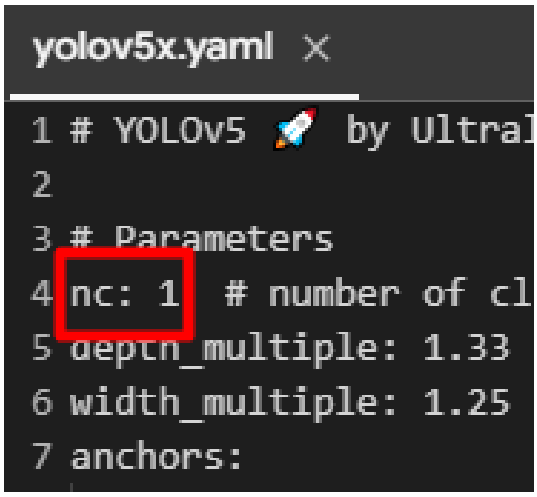


Figura 21

Configuración del archivo yolov5x.yaml

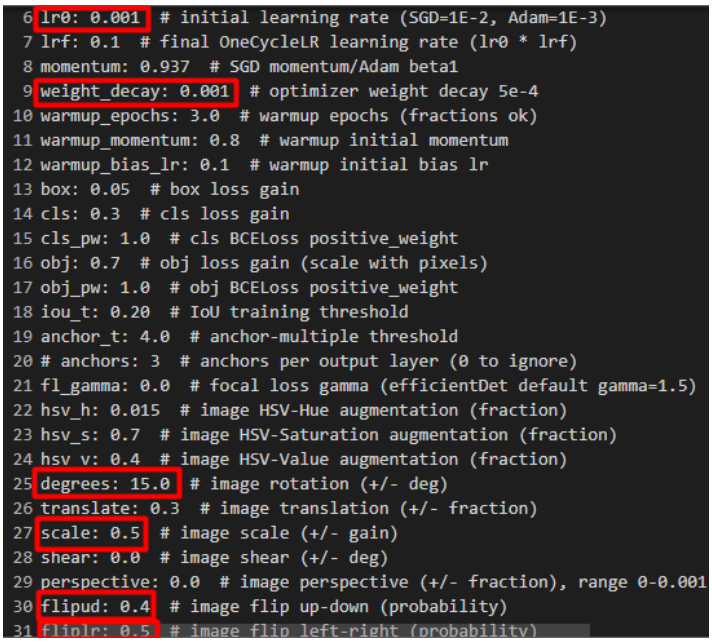


Ahora se procede a realizar una técnica llamada ajuste de hiperparámetros, que consiste en configurar el archivo donde se encuentran los hiperparámetos para entrenamiento del modelo, con el fin de optimizar y mejorar el rendimiento del modelo conforme pasan las épocas de entrenamiento evitando lo máximo posible el sobreajuste. Para el entrenamiento de los 3 modelos de este proyecto se realizaron los mismos ajustes.

Los hiperparámetros modificados fueron la taza de aprendizaje y el decaimiento de los pesos, aunque establecerlos en valor pequeño hace que el entrenamiento sea más lento, el aprendizaje del modelo se vuelve más estable y evita que se genere un sobreajuste al aprender demasiado rápido los datos de entrenamiento. Los demás hiperparámetros ajustados, son valores de técnicas de aumento de datos que tiene el mismo modelo para generar mayor cantidad de datos durante el entrenamiento. En concreto se modificaron 4 valores: la rotación de imágenes en 15°, el valor de la ganancia de escalado de la imagen, y el porcentaje de probabilidad de que las imágenes se volteen para arriba y para abajo. En la Figura 22 se pueden apreciar estos ajustes.

Figura 22

Ajuste de hiperparámetros



Una vez realizadas todas estas configuraciones, se procede a realizar el entrenamiento del modelo. Para ello se usa el programa train.py, donde se establecen algunos parámetros para empezar el entrenamiento. Para el caso de YOLOv5 y las dos versiones que se usaron, se tienen dos configuraciones distintas.

- Para el caso de Yolov5s se asigna un tamaño de imagen de 640x640, un tamaño de lote de 16, y un total de 100 épocas de entrenamiento, el peso específico para esta versión que es 'yolov5s.pt' y su respectivo archivo de configuración del modelo, añadiendo el ajuste de hiperparámetros.

Figura 23

Entrenamiento de YOLOv5s

```
!python train.py --img 640 --batch 16 --epochs 100 --data /content/datasets/Deteccion-personas-4/data.yaml \
--cfg /content/yolov5/models/yolov5s.yaml --weights yolov5s.pt --name resultados --cache \
--hyp /content/yolov5/data/hyps/hyp.scratch-med.yaml
```

- Con YOLOv5x se usa un tamaño de imagen de 640x640, un tamaño de lote de 16, un total de 75 épocas de entrenamiento, su peso específico 'yolov5x.pt' y su archivo de configuración yolov5x.yaml. Así como los hiperparámetros modificados anteriormente.

Figura 24

Entrenamiento de YOLOv5x

```
!python train.py --img 640 --batch 16 --epochs 75 --data /content/datasets/Deteccion-personas-4/data.yaml \
--cfg /content/yolov5/models/yolov5s.yaml --weights yolov5x.pt --name resultados --cache \
--hyp /content/yolov5/data/hyps/hyp.scratch-med.yaml
```

Una vez finalizado el entrenamiento se ejecuta la herramienta de tensorboard la cual permitirá monitorear el rendimiento del modelo y así comparar resultados.

Figura 25

Ejecución de tensorboard

```
# Start tensorboard
# Launch after you have started training
# logs save in the folder "runs"
%load_ext tensorboard
%tensorboard --logdir runs
```

Finalmente se exporta el modelo entrenado en formato Pytorch.

Figura 26

Exportación del modelo entrenado con YOLOv5

```
from google.colab import files
files.download('./runs/train/exp/weights/best.pt')
```

3.2.12. Entrenamiento de la red neuronal con YOLOv7

Para el entrenamiento con esta versión de YOLO, el proceso es prácticamente similar al que se realiza con YOLOv5, simplemente cambian algunos parámetros y comandos. Como primer paso se debe realizar la instalación de los requerimientos y dependencias clonando el repositorio oficial del modelo.

Figura 27

Instalación de requerimientos y clonación del repositorio de YOLOv7

```
# Download YOLOv7 repository and install requirements
!git clone https://github.com/WongKinYiu/yolov7
%cd yolov7
!pip install -r requirements.txt
```

Luego de esto se realiza la instalación de la librería roboflow, y si importa el dataset creado anteriormente en la plataforma Roboflow, pero exportado en formato YOLOv7 Pytorch.

Figura 28

Instalación de Roboflow e importación del dataset creado.

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="[redacted]")
project = rf.workspace("sistema-deteccin-de-personas").project("deteccion-personas-by0jw")
dataset = project.version(4).download("yolov7")
```

Una vez importado el dataset, se descarga un modelo pre-entrenado o también llamado checkpoint el cual ya fue entrenado anteriormente y se usa como punto de partida durante el entrenamiento, para que el modelo no tenga que aprender desde 0 todos los datos de entrenamiento.

Figura 29

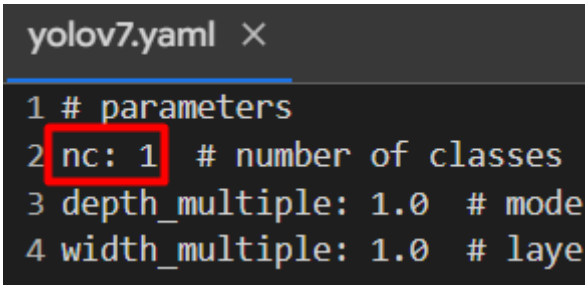
Descarga del checkpoint para el entrenamiento del modelo

```
%cd /content/yolov7
!wget https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7_training.pt
```

Como se hizo en los entrenamientos con las dos versiones del modelo YOLOv5, con este modelo se realizó las mismas configuraciones tanto del archivo de configuración del modelo, como del archivo de los hiperparámetros. Para el ajuste de hiperparámetros revisar la Figura 22 del documento.

Figura 30

Configuración del archivo YOLOv7.yaml



Después de haber realizado todas estas configuraciones previas, se realiza el entrenamiento del modelo usando el script train.py, estableciendo cada uno de los parámetros de entrenamiento, como número de lotes en 16, el número de épocas en 60, el checkpoint, y los dos archivos de configuración.

Figura 31

Entrenamiento del modelo YOLOv7

Como pasos finales, se ejecuta la herramienta tensorboard para poder monitorear el rendimiento que tuvo el modelo durante el entrenamiento, y con los conjuntos de validación. Que se puede revisar el comando en la Figura 25, así como la exportación del modelo en la Figura 26.

3.3. Programación

3.3.1. Programa del sistema de detección de personas

Para desarrollar el software de detección de personas, se comienza por la importación de cada una de las librerías necesarias, que fueron especificadas dentro de la Tabla 14.

Figura 32

Importación de librerías

```
import telepot
import subprocess
import urllib3
import time
import cv2
import torch
from models.experimental import attempt_load
from utils.general import non_max_suppression, scale_coords
from utils.datasets import letterbox
from utils.plots import plot_one_box
```

Nota. Autoría propia

Luego se importa el modelo, y la API KEY que permite la conexión con el chatbot de telegram. Después se define la función handle la cual contiene toda la funcionalidad del sistema en sí, ya que dentro de esta función se espera que se recepte el comando '/foto' desde el chat de telegram y una vez se recepte ese comando, el sistema activará la cámara para realizar la adquisición de la imagen.

Figura 33

Función que controla el sistema de detección de personas

```
def handle(msg):
    content_type, chat_type, chat_id = telepot.glance(msg)
    if content_type == 'text':
        # Si el mensaje es el comando /foto se ejecuta el sistema
        if msg['text'] == '/foto':
            # Captura una imagen con libcamera-still y la guarda en un archivo temporal
            subprocess.run(["libcamera-still", "-o", "/home/juan/Downloads/deteccion.jpg"])
```

Nota. Autoría propia.

Una vez la cámara toma la foto, y es guardad en un directorio, el sistema abre la imagen y la redimensiona a un tamaño de 640x640 pixeles, se la convierte a tensor y se normaliza para realizar la detección de personas aplicándole la supresión no máxima (NMS) con una confianza de 0.5 y un umbral IOU de 0.45. En la Figura 34 se puede observar lo descrito.

Figura 34

Sección del programa que realiza la detección de personas

```
imagen = cv2.imread('/home/juan/Downloads/Download/deteccion.png')
imagen_original = imagen.copy()  # Guardar una copia de la imagen original

# Preprocesar la imagen
imagen = letterbox(imagen, new_shape=(640, 640))[0]
# Convertir la imagen a tensor y normalizarla
tensor_imagen = torch.from_numpy(imagen).float() / 255.0
tensor_imagen = tensor_imagen.permute(2, 0, 1).unsqueeze(0)

# Realizar la detección de personas
with torch.no_grad():
    pred = modelo(tensor_imagen, augment=False)[0]

# Aplicar la supresión no máxima
pred = non_max_suppression(pred, 0.5, 0.45, classes=0, agnostic=False)
```

Después de realizar la detección se envía la imagen donde se ha realizado la inferencia, en la cual se encuentran dibujadas las cajas delimitadoras de cada persona que aparece en la imagen y dependiendo del resultado de la detección se envían los mensajes de alerta al chat de telegram.

Figura 35

Sección del programa que realiza el envío de la imagen y la alerta a telegram.

```
with open('/home/juan/Downloads/deteccion.jpg', 'rb') as photo:
                bot.sendPhoto(chat_id, photo)

if num_personas == 1:
    enviar_mensaje_con_reintentos(chat_id, f"*****ALERTA***** \n Se detectó 1 persona")
elif num_personas > 1:
    enviar_mensaje_con_reintentos(chat_id, f"*****ALERTA***** \n Se detectaron {num_personas}
elif num_personas == 0:
    enviar_mensaje_con_reintentos(chat_id, f"No se han encontrado personas")
```

Esto es solo una explicación de las partes más importantes del programa, para mejor entendimiento el código completo se encuentra en el Anexo B.

CAPÍTULO IV

RESULTADOS Y ANÁLISIS

Dentro de este capítulo, se lleva a cabo la evaluación del rendimiento de los tres modelos entrenados para la detección de personas, con el objetivo de determinar cuál es el más apropiado para este sistema y verificar su funcionamiento. Estas evaluaciones se realizan a través de pruebas de implementación en el laboratorio de fibra óptica de la FICA

4.1. Evaluación de rendimiento en los modelos entrenados

Para poder evaluar el rendimiento de cada uno de los modelos entrenados se ha utilizado la herramienta tensorboard que permite visualizar las gráficas del rendimiento de cada modelo conforme pasan las épocas de entrenamiento. Para esto, se toman en cuenta algunos parámetros como: Precisión, Recall, mAP, y la pérdida. Finalmente, para determinar cuál es el modelo más idóneo para este sistema, se analiza la matriz de confusión de cada modelo, y se realiza una inferencia en distintos escenarios para comprobar su precisión en la detección de personas.

4.1.2. Gráficas de mAP (Mean Average Precision)

La métrica mAP ofrece una evaluación global del desempeño de un modelo de detección de objetos al tener en cuenta la exactitud de las predicciones en distintas categorías y niveles de confianza, empleando umbrales específicos de superposición, como el IoU (Intersección sobre Unión).

Figura 36

Gráfica de la precisión mAP del modelo YOLOv5s

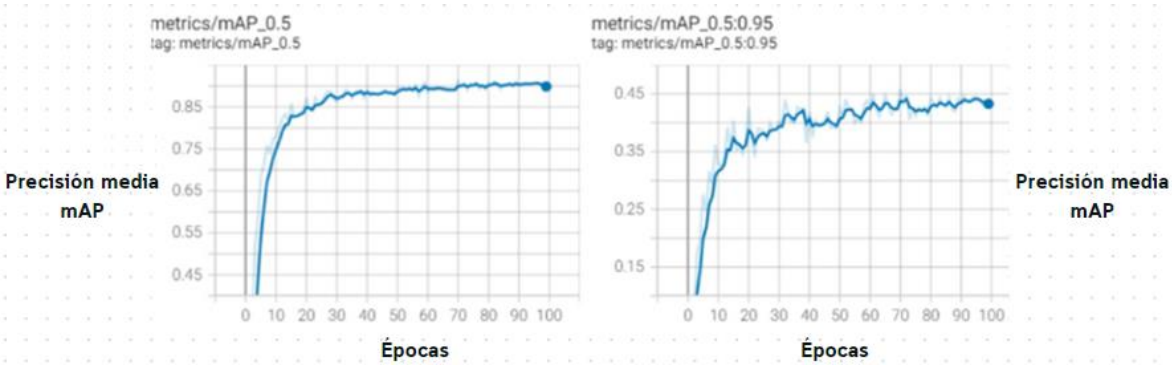


Figura 37

Gráfica de la precisión mAP del modelo YOLOv5x.

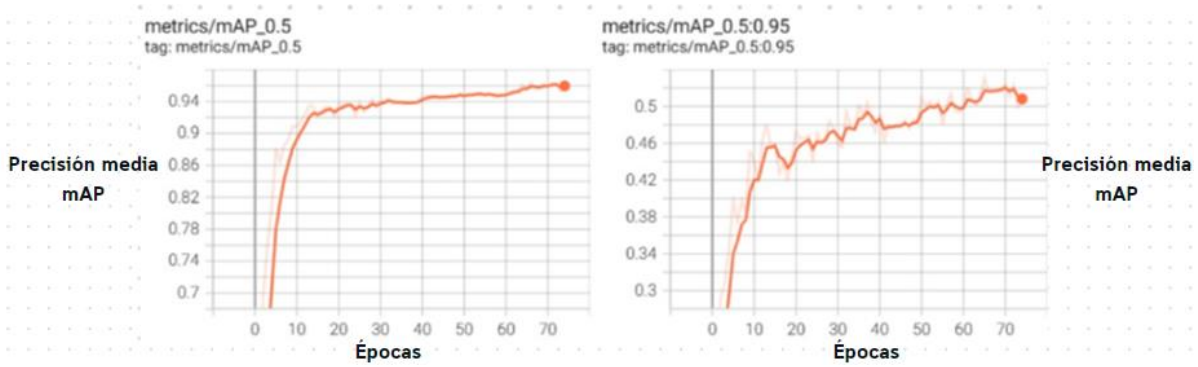
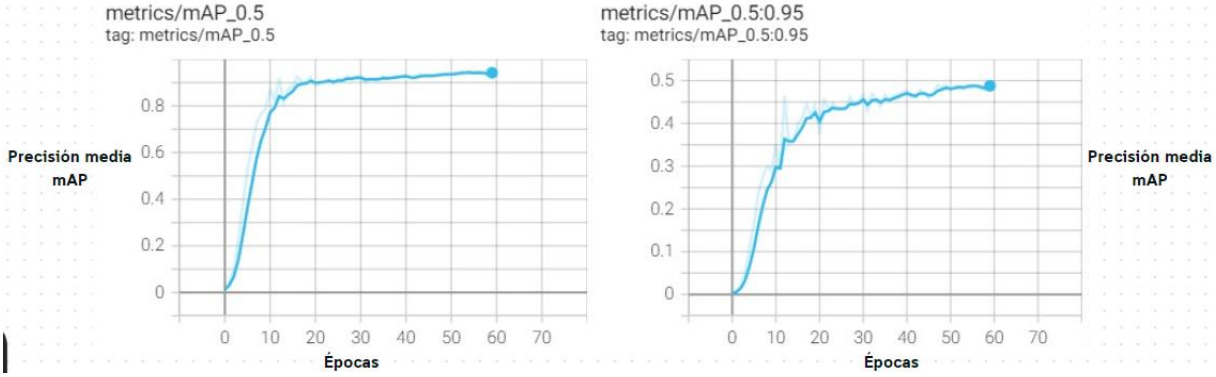


Figura 38

Gráfica de la precisión mAP del modelo YOLOv7



Al analizar cada una de estas gráficas se pueden observar dos parámetros, el del mAP_0.5 y el mAP_0.5:0.95, la notación mAP_0.5 se refiere a la precisión promedio media calculada utilizando un umbral de IoU (Intersección sobre Unión) del 50%. El IoU se utiliza para medir la superposición entre la caja delimitadora predicha por el modelo y la caja delimitadora real del objeto en la imagen. Un umbral de IoU del 50% significa que se considera que hay una detección correcta si la superposición entre las cajas es al menos del 50%. Por otro lado, mAP_0.5:0.95 significa que se calcula la precisión promedio media utilizando un rango de umbrales de IoU, específicamente desde 0.5 hasta 0.95 con incrementos de 0.05. Esto proporciona una medida más completa de la capacidad del modelo para hacer detecciones precisas a diferentes niveles de superposición entre las cajas delimitadoras predichas y reales.

Al comparar las gráficas de cada uno de estos parámetros en cada uno de los modelos, se puede observar que el modelo YOLOv5s y el modelo YOLOv7 son muy similares ya que alcanzan a superar el valor de 0.85, aunque el modelo YOLOv5x es el que más alto llega alcanzando más de 0.95, esto en cuanto refiere al mAP_0.5. Mientras que para el mAP_0.5:0.95 el que menor precisión tiene es el modelo YOLOv5s, seguido por el modelo YOLOv7, y finalmente el que más precisión tiene es el modelo YOLOv5x. Cabe recalcar que, el hecho de que un modelo tenga un valor de precisión mAP más alto no significa que sea mejor, ya que por ejemplo de los 3, el modelo YOLOv7 es el más estable en cuanto a esta métrica, porque como se observa los otros dos modelos tiene más picos o deformaciones en las gráficas.

4.1.3. Gráficas de precisión

La precisión es una métrica que analiza la capacidad de un modelo para hacer predicciones precisas en relación con el número total de predicciones realizadas. En

palabras simples, representa la proporción de predicciones correctas respecto al total de predicciones. Un alto valor de precisión señala que el modelo está acertando al clasificar las instancias, mientras que una baja precisión indica que el modelo está cometiendo errores en sus predicciones.

Figura 39

Gráfica de la precisión del modelo YOLOv5s

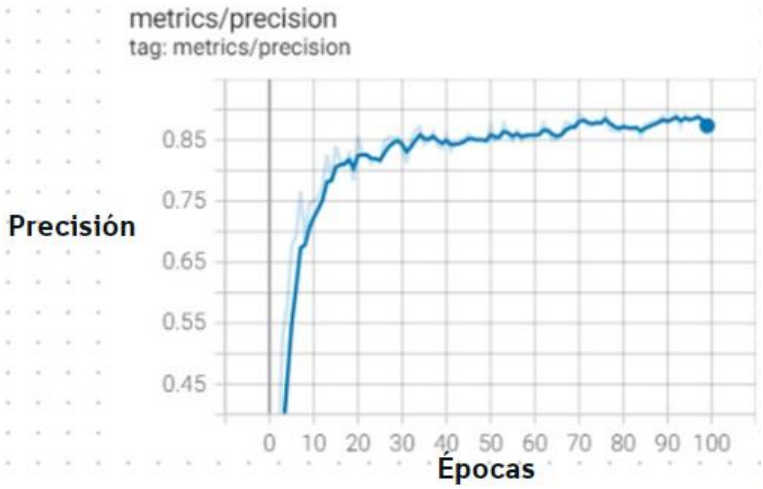


Figura 40

Gráfica de la precisión del modelo YOLOv5x

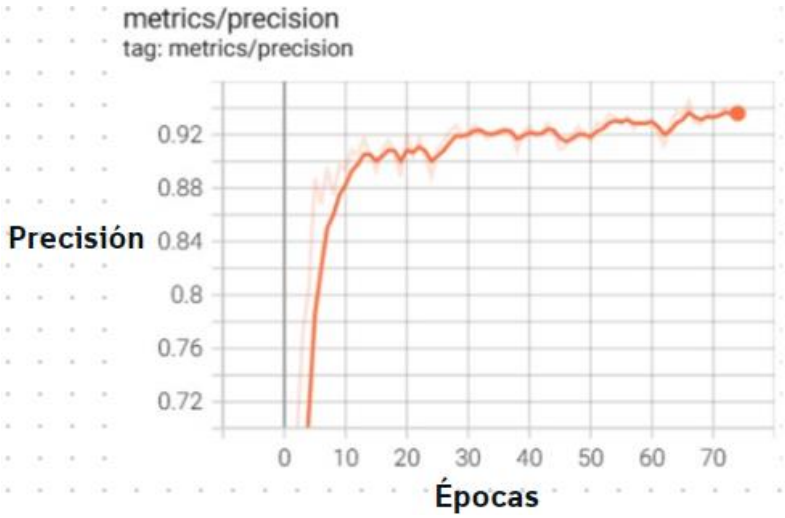
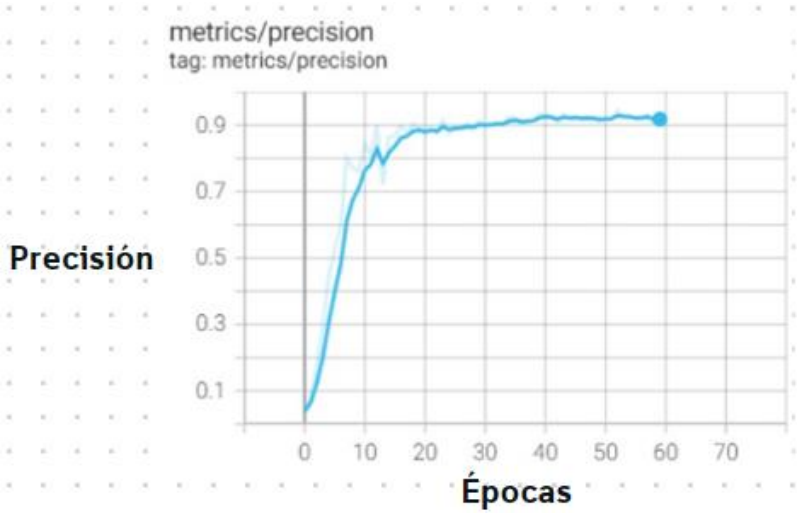


Figura 41

Gráfica de la precisión del modelo YOLOv7



Las tres gráficas muestran la precisión de cada modelo durante el entrenamiento, en donde se puede diferenciar claramente que el modelo que mayor precisión tiene es el modelo YOLOv5x ligeramente por encima del modelo YOLOv7, los dos superando el 0.9 de precisión. El que menos precisión tiene de los 3 es el modelo YOLOv5s.

4.1.4. Gráficas de recall

El recall se refiere a la capacidad del modelo para identificar y capturar la mayoría de las instancias de un objeto específico en una imagen. Un recall alto significa que el modelo está siendo efectivo al detectar la presencia de ese objeto, minimizando la cantidad de casos positivos que pasan desapercibidos.

Figura 42

Gráfica del recall del modelo YOLOv5s

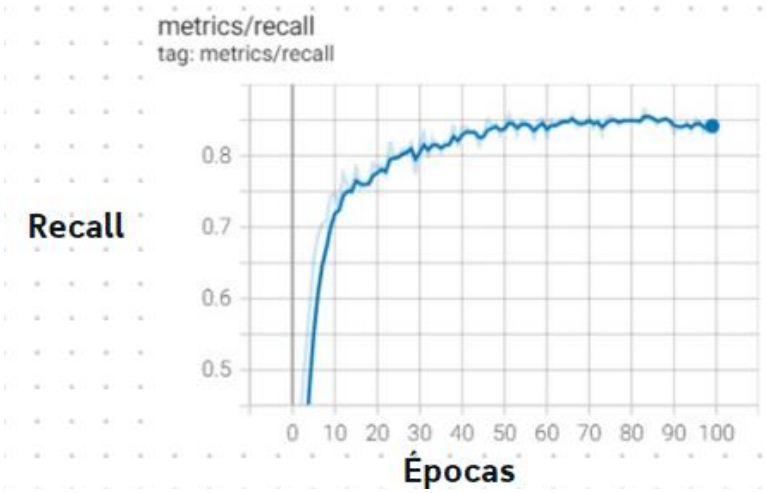


Figura 43

Gráfica del recall del modelo YOLOv5x

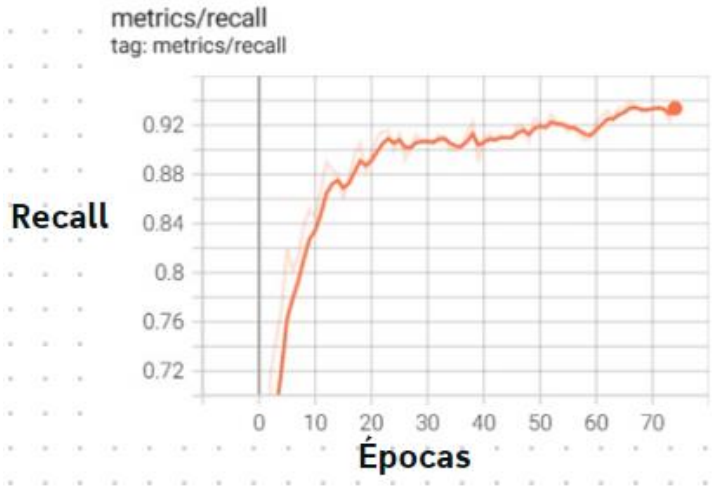
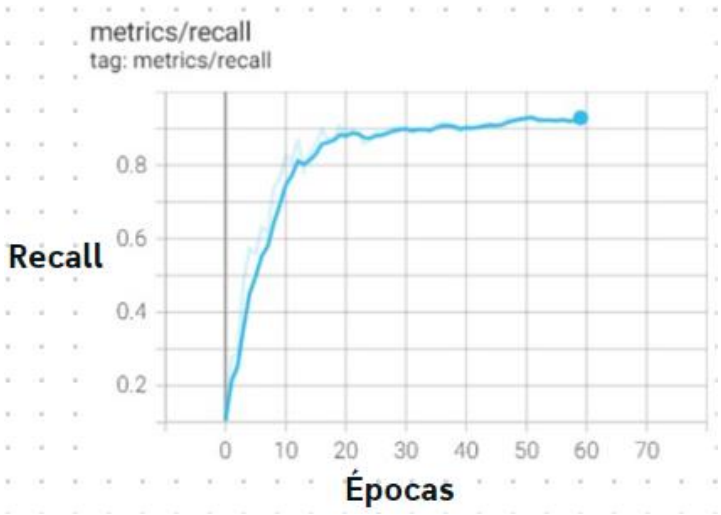


Figura 44

Gráfica del recall del modelo YOLOv7



Algo a tomar en cuenta para este análisis es que tanto la precisión como el recall van de la mano, y un indicio de que el modelo se está entrenando bien es que los valores de recall y precisión sean lo más parecidos posible. Y como se puede observar en las gráficas es que estas dos métricas se asemejan bastante. En cuanto a las gráficas, nuevamente el modelo que mayor recall tiene es el YOLOv5x, seguido del modelo YOLOv7, este último, demuestra ser más estable en cuanto al desempeño dentro de la gráfica.

4.1.5. Gráficas de pérdida

La pérdida es una métrica que evalúa qué tan imprecisas son las predicciones del modelo en relación con las etiquetas reales de los datos. Juega un papel crucial en la

fase de optimización durante el entrenamiento del modelo, ya que se emplea para modificar los pesos de la red neuronal con el fin de perfeccionar su habilidad predictiva.

Figura 45

Gráfica de la pérdida del modelo YOLOv5s

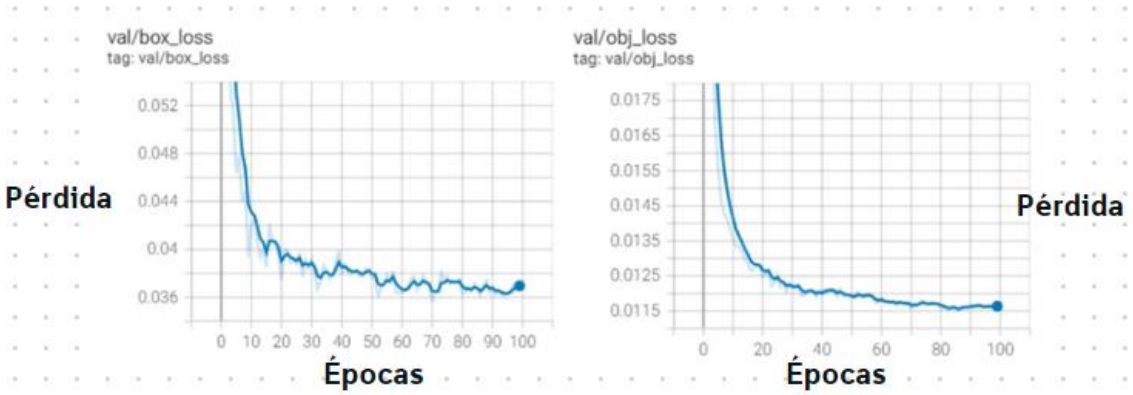


Figura 46

Gráfica de la pérdida del modelo YOLOv5x.

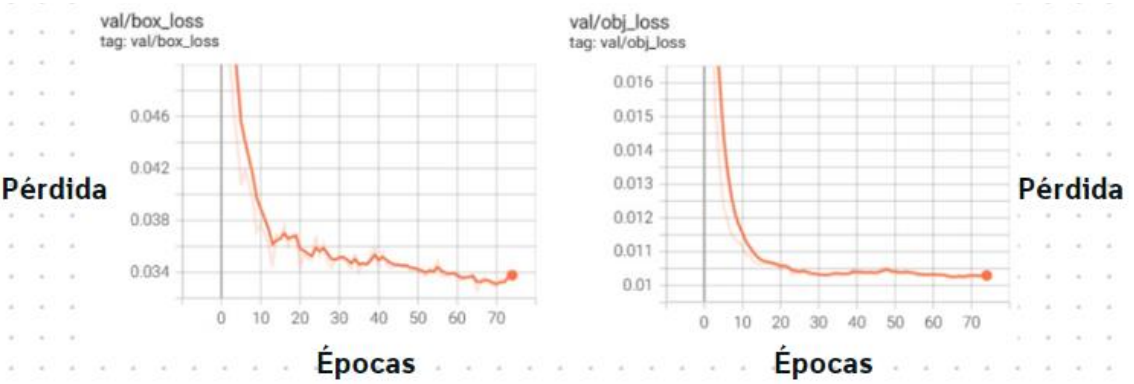
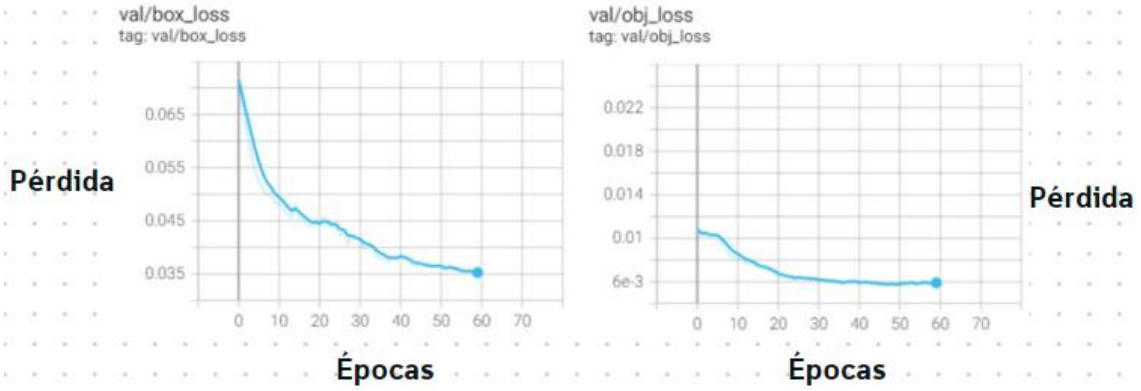


Figura 47

Gráfica de la pérdida del modelo YOLOv7



En todas las gráficas presentadas anteriormente se pueden observar dos métricas, la pérdida de caja, y la pérdida de objeto, la primera se refiere a la contribución de la red neuronal a la imprecisión en la predicción de las coordenadas de las cajas delimitadoras que rodean a los objetos en una imagen. Esta pérdida mide cuánto difieren las coordenadas predichas de las coordenadas reales de las cajas de los objetos. La segunda está relacionada con la confianza del modelo en la detección de un objeto en una ubicación específica. Mide cuánto se desvía la confianza predicha del modelo con respecto a la confianza real (es decir, si hay o no un objeto en la caja predicha).

Analizando los comportamientos de las pérdidas para cada modelo, se observa que tienen una tendencia a decrecer que es lo que se espera, la diferencia entre los 3 modelos es notable en cuanto a los valores de pérdida que van obteniendo conforme pasan las épocas, en este caso el que tiene valores más bajos es el modelo YOLOv7, llegando a tener un valor de pérdida de caja de 0.035 y una pérdida de objeto de 0.006. Determinando así que el modelo YOLOv7 es el que menos imprecisión tiene a la hora de realizar predicciones. En la siguiente tabla se puede apreciar la comparación de los 3 modelos

durante el entrenamiento, tomando en cuenta el número de capas, número de parámetros y su porcentaje de precisión con el conjunto de datos de entrenamiento.

Tabla 15

Tabla de ablación de comparación de los 3 modelos

Modelo	N° de capas	N° Parámetros	Rendimiento	Tiempo
YOLOv5s	214	7022326	92% de precisión	38 min.
YOLOV5x	445	86217814	94% de precisión	1,4 hrs
YOLOv7	415	37196556	96% de precisión	1,09 hrs

La tabla indica que tan robusto es cada modelo en cuanto a su arquitectura, demostrando que el modelo más pesado de los 3 es el modelo YOLOv5x, lo cual lo hace un buen modelo, el modelo más preciso de los 3 durante este entrenamiento fue el modelo YOLOv7 con un 96% de precisión, este valor fue obtenido del entrenamiento del modelo y su rendimiento con el conjunto de datos de validación. Lo que también demuestra cómo ha evolucionado cada modelo en su arquitectura llegando a un modelo YOLOv7 mucho más optimizado para realizar detecciones precisas en un menor tiempo.

4.2. Pruebas

Para comprobar en la práctica, cuál es el modelo que mejor funciona para la detección de personas, se realizaron algunas pruebas, en varios escenarios que permitan determinar qué modelo fue más preciso en estas pruebas.

4.2.1. Pruebas con el aula llena

En las Figuras 48, 49 y 50, se muestran las pruebas realizadas con cada uno de los modelos, en un escenario donde el aula de clase esté llena o donde hay bastantes personas para realizar la detección. En las imágenes tomadas se

encuentran 14 personas donde se comprobará cuantas personas son capaces de detectar cada modelo.

Figura 48

Inferencia realizada con YOLOv5s para el escenario 1

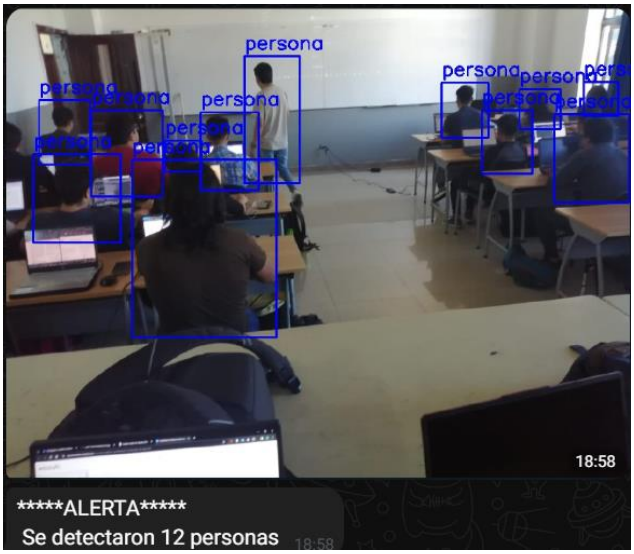
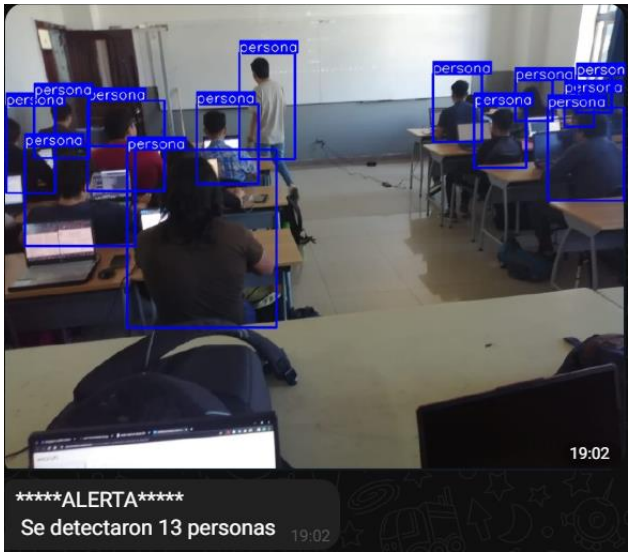


Figura 49

Inferencia realizada con YOLOv5x para el escenario 1

Figura 50

Inferencia realizada con YOLOv7 para el escenario 1



En las inferencias realizadas con los 3 modelos para este escenario, se tiene que el modelo YOLOv5s ha detectado 12 personas, el modelo YOLOv5x ha detectado 10, y el modelo YOLOv7 ha logrado detectar 13 personas de la imagen. Todos 3 no han tenido ningún falso positivo como resultado de la detección de personas.

4.2.2. Pruebas con el aula medio llena

En las siguientes figuras se evidencian las pruebas realizadas dentro del escenario de un aula medio llena, para comprobar que tan precisos son los modelos.

Figura 51

Inferencia realizada con YOLOv5s para el escenario 2

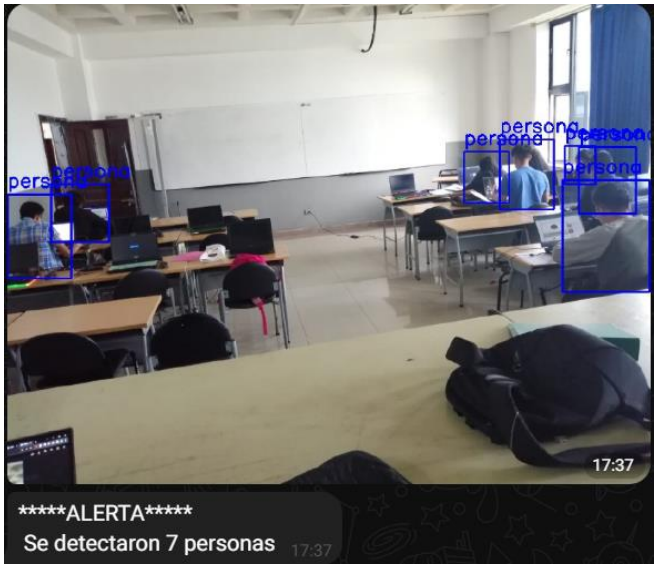


Figura 52

Inferencia realizada con YOLOv5x para el escenario 2

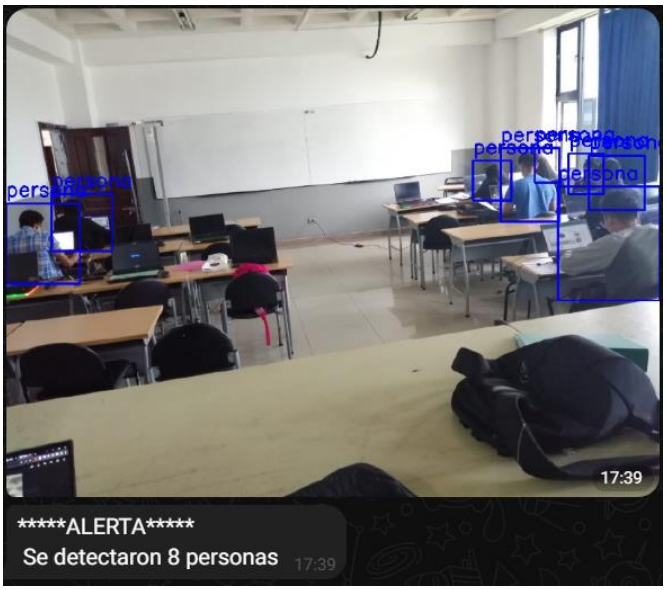
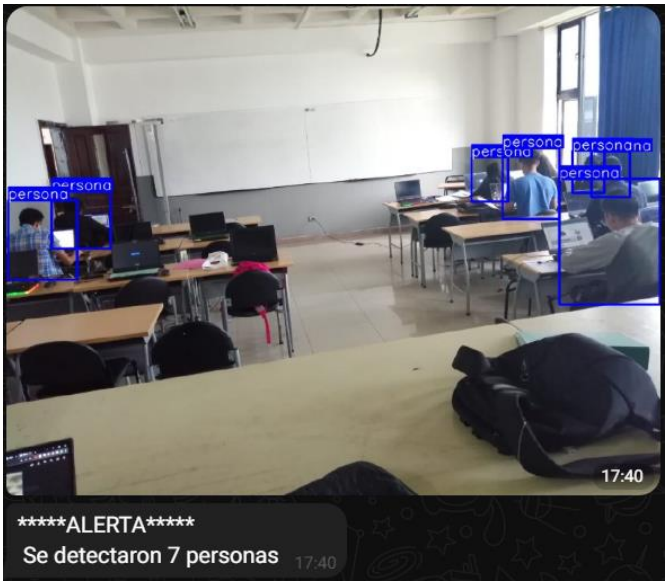


Figura 53

Inferencia realizada con YOLOv7 para el escenario 2



Al realizarse la inferencia en este escenario, donde el aula no está completamente llena, se tiene un total de 8 personas en la imagen, y es posible visualizar que tanto el modelo YOLOv5s y el modelo YOLOv7 logran detectar 7 personas, pero, el modelo YOLOv5x detecta 8 personas en la imagen, además de que, en las pruebas realizadas en este escenario, tampoco se encontraron falsos positivos.

4.2.3. Pruebas en aula con pocas personas

Se realizaron pruebas con el aula casi vacía, un entorno donde hay pocas personas y así demostrar que cada modelo ha sido capaz de generalizar diferentes entornos para detectar personas.

Figura 54

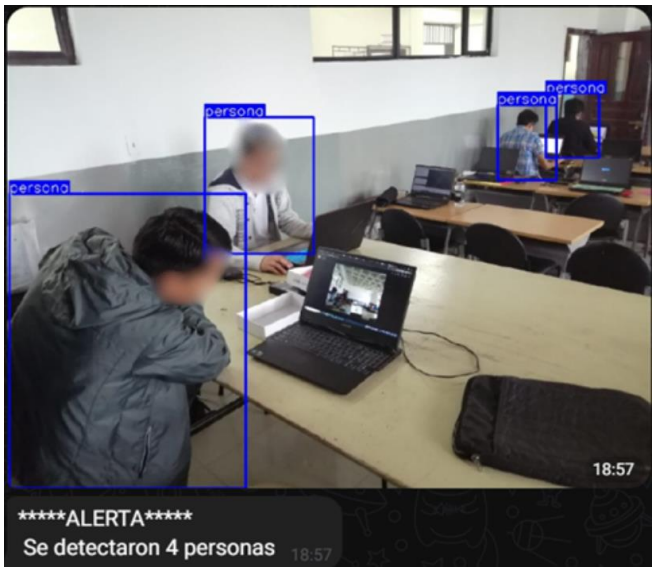
Inferencia realizada con YOLOv5s para escenario 3

Figura 55

Inferencia realizada con el modelo YOLOv5x en escenario 3

Figura 56

Inferencia realizada con el modelo YOLOv7 en escenario 3



Para este escenario, se obtuvo como resultado que los 3 modelos fueron capaces de detectar a todas las personas en la imagen tomada y nuevamente ninguno de los 3 detectó falsos positivos.

4.2.4. Pruebas con el aula vacía

En este caso se realizaron pruebas en un escenario donde el aula esté vacía, esto se realiza con el fin de comprobar si los modelos han generalizado bien y son capaces de reconocer que en ese momento no hay personas, en otras palabras, se realizan estas pruebas para demostrar si existen falsos positivos durante la detección. Para este escenario se puso como trampa, una mochila, y en el fondo del aula hay un mandil colgado para comprobar si el sistema logra identificarlos como persona o no.

Figura 57

Inferencia realizada con el modelo YOLOv5s en escenario 4

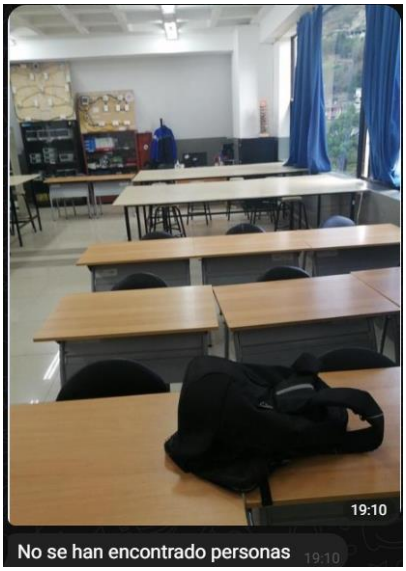


Figura 58

Inferencia realizada con el modelo YOLOv5x en el escenario 4

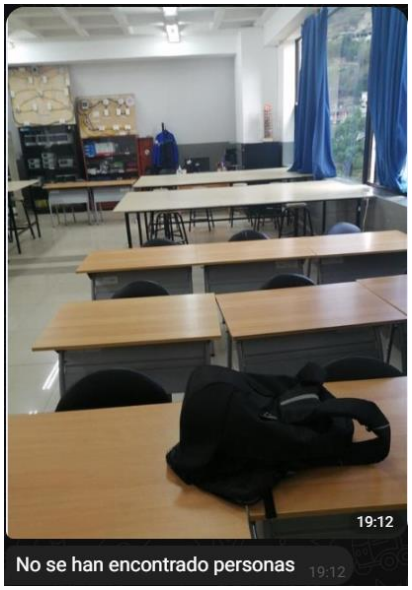
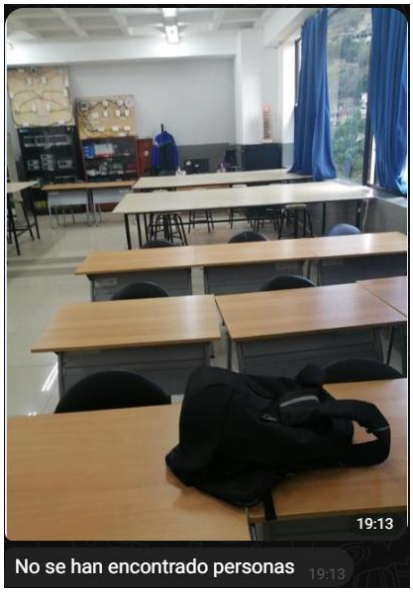


Figura 59

Inferencia realizada con el modelo YOLOv7 en el escenario 4



Como se puede comprobar en cada una de las imágenes, ninguno de los 3 modelos detectó falsos positivos, lo que indica que los 3 son capaces de reconocer cuando no hay personas en un escenario así.

4.2.5. Prueba en aula con poca iluminación

Finalmente, se realizaron pruebas en un ambiente donde tenga cambios de iluminación como en este escenario en el cual el sistema demostrar que también logra realizar detecciones en lugares con poca luz.

Figura 60

Inferencia realizada con el modelo YOLOv5s en el escenario 5

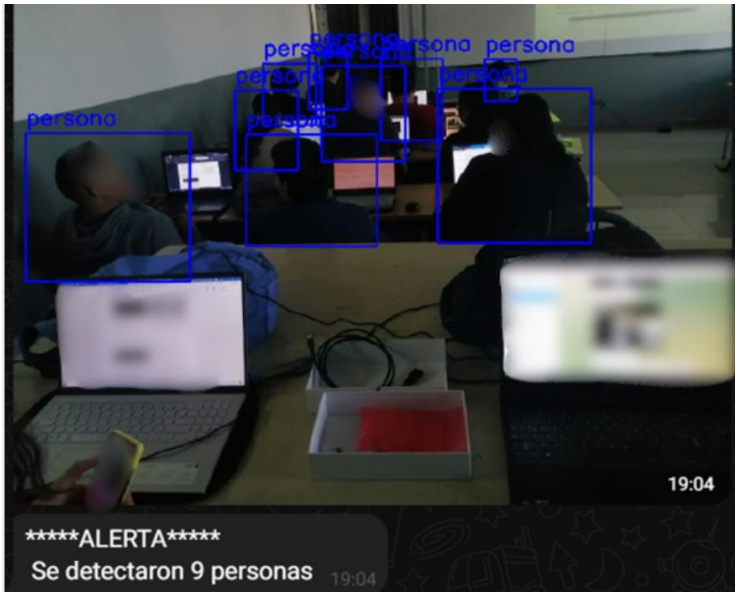


Figura 61

Inferencia realizada con el modelo YOLOv5x en el escenario 5

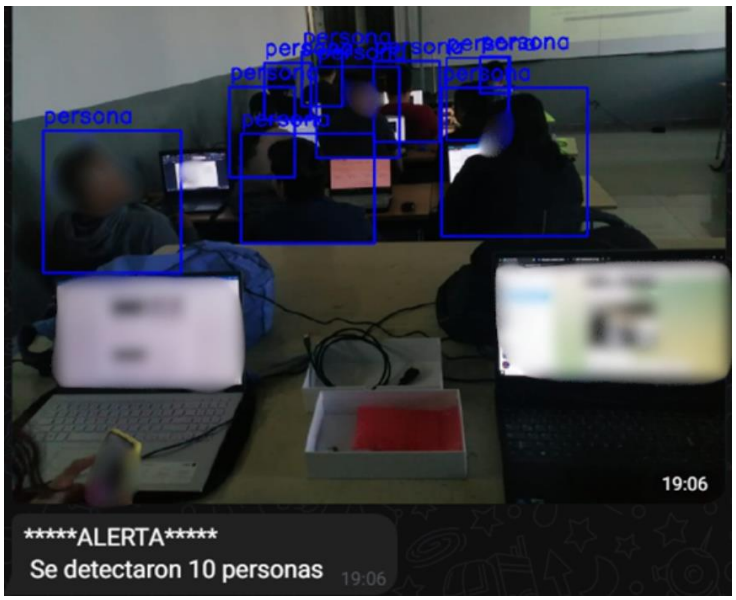
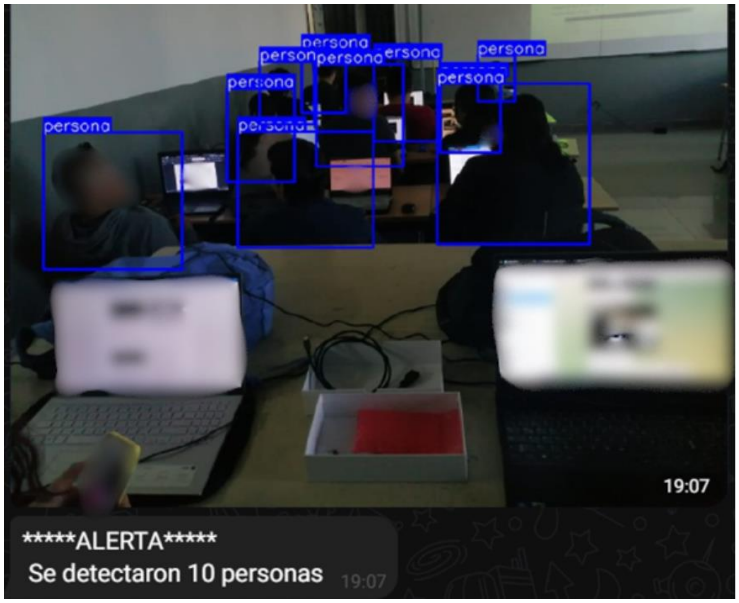


Figura 62

Inferencia realizada con el modelo YOLOv7 en el escenario 5



Las pruebas realizadas en este escenario muestran que el modelo YOLOv5s es el que no logró detectar a todas las personas, mientras que los modelos YOLOv5x y YOLOv7 lograron detectar las 10 personas que se encuentran en la imagen tomada por la cámara. Ningún modelo realizó detecciones con falsos positivos.

4.2.6. Pruebas con el sistema implementado

Además de realizar las pruebas en distintos escenarios para comprobar que tan bien funciona el sistema con cada modelo, se realizaron pruebas con el sistema ya implementado tomando en cuenta el diseño realizado previamente donde va a estar ubicada la cámara.

Figura 63

Prueba con modelo YOLOv5s

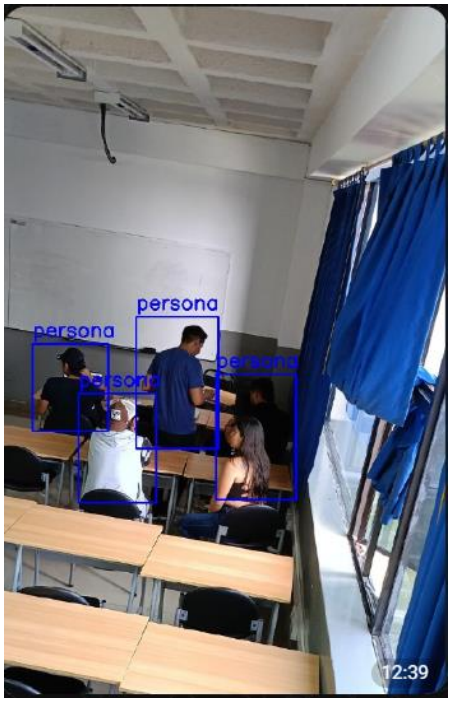


Figura 64

Pruebas con modelo YOLOv5x

Figura 65

Pruebas con modelo YOLOv7

4.3. Comparaciones finales para la elección del modelo

Una vez realizadas todas estas pruebas con cada uno de los modelos entrenados, se tabularon los datos tanto de precisión, como de tiempo de inferencia, para poder determinar y elegir cuál es el modelo que se va a utilizar en el sistema de detección de presencia de personas. En la Tabla 16 se realiza la comparación en cuanto a precisión, y en la Tabla 17 se comparan los tiempos de inferencia de cada modelo.

Tabla 16

Tabla de ablación de precisión durante las pruebas de inferencia realizadas

Modelo	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Precisión promedio (%)
YOLOv5s	85,70%	87,50%	100%	100%	100%	94,64%
YOLOv5x	71,40%	100%	100%	100%	100%	94,28%
YOLOv7	92,90%	87,50%	100,00%	100%	100%	96,08%

En cada prueba realizada se obtuvo un porcentaje de precisión de cada modelo, el valor obtenido es el resultado de la siguiente fórmula:

$$Precisión = \frac{Verdaderos\ positivos}{Verdaderos\ positivos + Falsos\ negativos}$$

En donde Verdaderos positivos es la cantidad de detecciones correctas que ha realizado el modelo, y Falsos negativos es la cantidad de personas no detectadas por el modelo, ya que, al no haber falsos positivos, se toman en cuenta los falsos negativos en su lugar. Se logra demostrar que la precisión entre los modelos YOLOv5s y YOLOv5x es bastante pareja, pero que, por una pequeña diferencia, el modelo YOLOv5s es quien demuestra mejor rendimiento entre los dos. Por otro lado, el modelo que mejor precisión demostró de los 3 modelos es YOLOv7, que con 96% de precisión demuestra ser el más idóneo para esta tarea en cuanto a precisión se refiere.

Tabla 17

Tabla de ablación del tiempo de inferencia entre los 3 modelos

Modelo	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Tiempo promedio (s)
YOLOv5s	6,17	7,67	6,94	6,65	7,41	6,968
YOLOv5x	26	24,51	24,57	24,92	25,05	25,01
YOLOv7	15,02	17,73	18,24	15,66	18,17	16,964

Al realizar las inferencias una métrica que también es bastante importante para este sistema es el tiempo que tarda cada modelo en realizar la detección, donde el modelo YOLOv5s es el que más rápido realiza las inferencias con un tiempo promedio de 6,968 segundos. El modelo que tarda más tiempo en esta comparación es el modelo YOLOv5x, con un tiempo de 25,01 segundos, esto es debido a que es un modelo bastante pesado.

Teniendo estas dos comparaciones se debe elegir qué modelo se debe usar para este sistema. Aunque la precisión puede definir cuál es el mejor modelo para detectar personas, también se debe de tomar en cuenta las métricas anteriores que fueron detalladas al principio de este capítulo. Si bien los modelos YOLOv5x y YOLOv7 son los más precisos, y Yolov5s es el más rápido en realizar la detección. ¿ Cuál es el modelo que se debe elegir?

La precisión es importantísima para este sistema ya que un falso positivo o negativo, puede perjudicar que se realice o no un rescate o evacuar a la persona durante la emergencia. Pero hay que tomar que el tiempo de inferencia también lo es, ya que cada segundo cuenta, y el tiempo que tarde el sistema en realizar la detección determinará que tan pronto se tomen las acciones requeridas para salvar la vida de una persona. Es por ello por lo que este sistema debe realizar la detección y enviar la alerta en menos de 1 minuto. Entonces lo que se busca es un balance entre precisión y tiempo, por lo tanto, el modelo

escogido para implementar este sistema de detección de presencia de personas es el modelo YOLOv7 por su buena precisión y su tiempo de inferencia.

4.4. Notificación y alerta del sistema de detección de presencia de personas

Una vez escogido el modelo con el cuál se va a trabajar e implementar el sistema, se necesita que se manden alertas al momento de que se detecte una persona dentro de un aula o laboratorio. Para ello lo primero que debe de realizar el encargado del personal de seguridad de la facultad, es ingresar el comando '/foto' dentro del chatbot de telegram como se observa en la siguiente figura.

Figura 66

Comando para iniciar el sistema de detección de presencia

Esto iniciará el sistema de detección de presencia, tomará una foto y analizará si hay personas adentro o no las hay, y enviará una notificación al celular indicando la cantidad de personas que hay en caso de haberlas, además de una imagen con la foto analizada por el sistema. En las figuras 67 y 68 se puede apreciar de mejor forma lo que se acaba de describir.

Figura 67

Notificación de la alerta del sistema de detección de presencia.

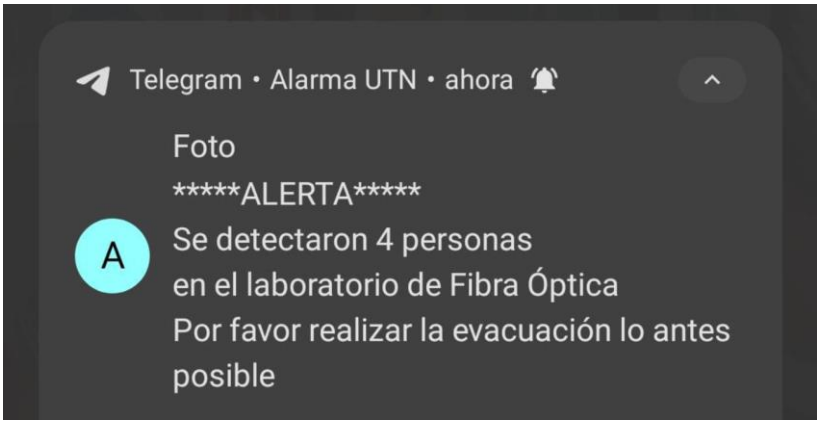
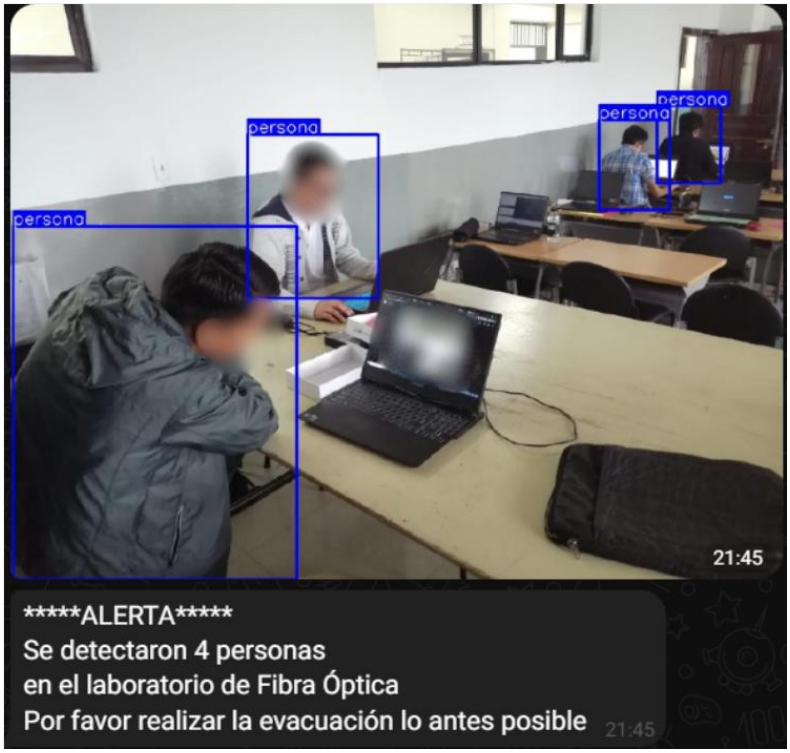


Figura 68

Visualización de la alerta dentro de la aplicación



Conclusiones y recomendaciones

Conclusiones

La implementación exitosa de un sistema embebido basado en visión artificial, con el modelo YOLO, para la detección de personas, representa un avance significativo en la seguridad y preparación para emergencias en la Universidad Técnica del Norte. Este proyecto proporciona una herramienta efectiva para la gestión de evacuaciones y operaciones de rescate en situaciones críticas. La conjunción de conocimientos teóricos sobre emergencias, sistemas embebidos y visión artificial ha culminado en una solución integral que no solo es técnica, sino también un compromiso palpable con la seguridad y el bienestar de la comunidad educativa.

La revisión del plan de emergencias de la universidad, junto con la exploración de temas relacionados con cada capa de la arquitectura del sistema, ha sentado las bases esenciales para este trabajo. La inclusión del deep learning y la visión artificial en el análisis teórico ha permitido comprender a fondo las tecnologías clave en la detección de personas. Este estado del arte muestra las capacidades transformadoras del deep learning en la mejora continua de la seguridad y la preparación para emergencias en la Universidad Técnica del Norte. La integración de estos conocimientos en el desarrollo del sistema no solo aprovecha el amplio campo de aplicación que tiene la visión artificial, sino que también enriquece la base teórica y técnica, abriendo nuevas posibilidades para futuros avances en la detección de personas en entornos universitarios.

La definición de requerimientos junto con el diseño de hardware y software, marcan un avance clave. La combinación de componentes de hardware y la implementación de modelos YOLOv5 y YOLOv7, respaldados por un sólido proceso de entrenamiento, promete una detección precisa y en poco tiempo. Esta fase sienta las bases

para la implementación, representando un paso importante en la capacidad del sistema para contribuir de manera efectiva a la seguridad y respuesta a emergencias en la universidad.

La construcción del sistema de detección de presencia, tomando en cuenta los requerimientos funcionales establecidos, posibilita la implementación coherente tanto del hardware como del software. Este enfoque ha dado lugar a una solución eficiente y confiable que responde de manera directa a las necesidades del Departamento de Seguridad y Gestión de Riesgos, estableciendo así un marco sólido para fortalecer la seguridad y la capacidad de respuesta ante situaciones críticas en la Universidad Técnica del Norte.

Las pruebas realizadas con cada uno de los modelos implementados determinan la elección del modelo YOLOv7 para usarlo en el sistema de detección de presencia debido a su gran precisión y generalización que tiene en distintos escenarios. Además de su rápido tiempo de inferencia lo que permite cumplir con los requerimientos establecidos, lo que logra una pronta respuesta por parte del departamento de seguridad en cuanto al rescate de una persona o su evacuación lo antes posible y evita cualquier riesgo que comprometa su integridad.

Recomendaciones

Este proyecto está enfocado exclusivamente para detección de personas, y dependiendo del enfoque y la necesidad, se debe realizar una comparación entre distintos modelos con la finalidad de determinar cuál es el mejor para la aplicación que se le va a dar. No todos los modelos son apropiados para las mismas tareas.

Para que un modelo quede bien entrenado y se evite el sobreajuste, es importante tener un dataset de imágenes bien etiquetado, que sea variado (de preferencia usar técnicas de aumento de datos), y utilizar técnicas como el ajuste de hiperparámetros, dropout, o early stopping, que permitirán un mejor rendimiento en el entrenamiento del modelo.

Si se desea tener un mejor rendimiento en cuanto a tiempo de inferencia y procesamiento, se recomienda utilizar otro tipo de sistemas embebidos que tengan componentes de hardware más robustos, o a su vez utilizar aceleradores USB, que permitan mejorar el rendimiento del sistema en cuanto a hardware se refiere ya que esta fue una gran limitante para este trabajo.

Si se opta por trabajar con sistemas embebidos como Raspberry Pi 4 modelo B, se recomienda utilizar un sistema operativo de 64 bits, ya que, al momento de trabajar con la arquitectura de 32 bits, se tuvieron muchos inconvenientes en cuanto a compatibilidad e instalación de librerías para trabajar con visión artificial.

Se recomienda realizar un análisis en cuanto a tener algún método de backup para cuando falle la alimentación energética, como por ejemplo usar alternativas, como bancos de poder o baterías, para que suministren la energía necesaria y así mantener el sistema encendido mientras se restablece la otra fuente de energía principal con la que se alimenta el sistema.

De igual forma, con la parte inalámbrica, se recomienda que además del protocolo IEEE 802.11 que es el protocolo de comunicación inalámbrica principal, utilizado para conectar los dispositivos, se tome en cuenta el uso de otro protocolo, como la implementación de módulos de redes móviles como puede ser un módulo GPRS o de una tecnología mucho más actual, con el fin de tener otro medio de comunicación inalámbrica en caso de que por una emergencia la red falle.

Referencias Bibliográficas

(2021). *PLAN DE EMERGENCIAS UTN*. Departamento de Seguridad y Gestión de Riesgos, Ibarra.

Universidad Técnica del Norte. (2022). Obtenido de https://www.utn.edu.ec/campus-universitarios/

Acharya, A. (2022, noviembre 7). *Guide to Image Segmentation in Computer Vision: Best Practices*. https://encord.com/blog/image-segmentation-for-computer-vision-best-practice-guide/

Arducam. (s. f.). Arducam IMX519 PDAF&CDAF Autofocus Camera Module for Raspberry Pi, NVIDIA® Jetson Nano/Xavier NX/AGX Orin/Orin Nano/Orin NX. *Arducam*. Recuperado 4 de noviembre de 2023, de https://www.arducam.com/product/imx519-autofocus-camera-module-for-raspberry-pi-arducam-b0371/

Asana. (2023, enero 8). *¿Quiénes son los stakeholders de un proyecto? Descubre cómo identificarlos y gestionarlos para asegurar el éxito del proyecto [2023] • Asana*. Asana. https://asana.com/es/resources/project-stakeholder

AV Electronics. (s. f.). Raspberry Pi 4 Model B 2GB. *AV Electronics*. Recuperado 29 de octubre de 2023, de https://avelectronics.cc/producto/raspberry-pi-4-model-b-2gb/

AWS. (2023). *¿Qué es la visión artificial? - Explicación de la IA y el aprendizaje automático de imágenes - AWS*. Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/computer-vision/

baeldung. (2022, septiembre 27). *What Is Content-Based Image Retrieval? | Baeldung on Computer Science*. https://www.baeldung.com/cs/cbir-tbir

Datacamp. (2022, septiembre). YOLO Object Detection Explained: A Beginner's Guide [Blog]. *YOLO Object Detection Explained*. https://www.datacamp.com/blog/yolo-object-detection-explained

Etecé. (2021). Emergencia—Concepto, tipos y emergencias médicas. *https://concepto.de/*. https://concepto.de/emergencia/

García, E. M. (2012). *Visión artificial*. https://ftp.isdi.co.cu/Biblioteca/BIBLIOTECA%20UNIVERSITARIA%20DEL%20ISDI/COLECCION%20DE%20LIBROS%20ELECTRONICOS/LE-1069/LE-1069.pdf

Gardner, K. (2021, diciembre 24). *Yolov5m architecture · Issue #6094 · ultralytics/yolov5*. GitHub. https://github.com/ultralytics/yolov5/issues/6094

Gobierno de México. (2023). *¿Qué es una emergencia?* https://embamex.sre.gob.mx/marruecos/index.php/es/emergencias-proteccionmx/que-es-una-emergencia

IBM. (s. f.). *What is Computer Vision? | IBM*. Recuperado 16 de octubre de 2023, de https://www.ibm.com/es-es/topics/computer-vision

J.L, B. (2023, junio 27). *Sistemas Embebidos*. ElectrónicaOnline. https://electronicaonline.net/electronica/sistemas-embebidos/

K, A. (2023, julio 20). What is Computer Vision Libraries and use cases of Computer Vision Libraries? *DevOpsSchool.Com*. https://www.devopsschool.com/blog/what-is-computer-vision-libraries-and-use-cases-of-computer-vision-libraries/

Klingler, N. (2023, enero 3). *Object Tracking in Computer Vision (2023 Guide)*. Viso.Ai. https://viso.ai/deep-learning/object-tracking/

Kundu, R. (2023, enero 17). *YOLO Algorithm for Object Detection Explained [+Examples]*. https://www.v7labs.com/blog/yolo-object-detection, https://www.v7labs.com/blog/yolo-object-detection

Luchetti, S. (2021, junio 1). Sistema embebido y sus características | Conceptos fundamentales. *Tribalyte Technologies*. https://tech.tribalyte.eu/blog-sistema-embebido-caracteristicas

Montenegro, B., Flores-Calero, M., Montenegro, B., & Flores-Calero, M. (2022). Detección de peatones en el día y en la noche usando YOLO-v5. *Ingenius. Revista de Ciencia y Tecnología*, *27*, 85-95. https://doi.org/10.17163/ings.n27.2022.08

Oracle. (2022, mayo 4). *Qué es PyTorch: Una guía completa*. https://developer.oracle.com/es/learn/technical-articles/what-is-pytorch

Patel, A. (2020, junio 11). What is Object Detection? *ML Research Lab*. https://medium.com/ml-research-lab/what-is-object-detection-51f9d872ece7

Raspberry Pi. (s. f.-a). *Buy a Raspberry Pi 4 Model B*. Raspberry Pi. Recuperado 29 de octubre de 2023, de https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

Raspberry Pi. (s. f.-b). *Raspberry Pi 4 Model B specifications*. Raspberry Pi. Recuperado 29 de octubre de 2023, de https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

Rodríguez, H. (2021, abril 28). *¿Qué es OpenCV y para qué sirve? 👁 [2021]*. https://www.crehana.com. https://www.crehana.com/blog/transformacion-digital/que-es-opencv/

S, I. G., & S, V. C. (2015). La visión artificial y los campos de aplicación. *Tierra Infinita*, *1*(1), Article 1. https://doi.org/10.32645/26028131.76

Solawetz, J. (2020, junio 29). *What is YOLOv5? A Guide for Beginners.* Roboflow Blog. https://blog.roboflow.com/yolov5-improvements-and-evaluation/

SuperAnnotate. (2023, mayo 30). *What is image classification? Basics you need to know | SuperAnnotate*. https://www.superannotate.com/blog/image-classification-basics

Ultralytics. (2022). *YOLOv7*. https://docs.ultralytics.com/models/yolov7

Weerasinghe, T. (2022, septiembre 21). *Yolov7 Architecture in Research Paper · Issue #762 · WongKinYiu/yolov7*. GitHub. https://github.com/WongKinYiu/yolov7/issues/762

ANEXOS

Anexo A. Entrevista realizada al Director del Departamento de Seguridad y Gestión de Riesgo

Nombre del entrevistado: Ing. Edwar Vásquez

Entrevistador: Juan José Vásquez

¿Cuál es el cargo que tiene dentro del DSGR?

Director del DSGR.

¿Qué funciones realiza dentro de su cargo?

Gestión administrativa del departamento.

¿Existen inconvenientes al momento de revisar si todas las personas han evacuado?

Si, hay inconvenientes ya que, si hay puertas que están cerradas o humo, es difícil realizar el barrido y determinar si hay alguien adentro.

¿Cree conveniente implementar un sistema de detección de presencia de personas para una emergencia?

Si, es conveniente implementar un sistema de detección de presencia, debido a que mejora la respuesta ante la emergencia y el rescate, esto se puede implementar principalmente laboratorios, para que les sirva de ayuda a los bomberos de materiales peligrosos, y así mejorar los tiempos de respuesta.

¿Si se implementa este sistema, como debería ser su funcionamiento?

Primero que todo, debe ser implementado en las áreas más peligrosas como laboratorios, se necesita realizar un análisis de riesgo para ello. La aplicación debe permitir visualizar las alertas al presidente del COE institucional, al Director del DSGR, y al jefe de seguridad.

¿Cuánto tiempo debe demorarse el sistema en detectar y enviar la alerta?

El tiempo de detección y envío de la alerta debe ser en menos de 1 minuto.

Anexo B. Código del sistema de detección de presencia de personas

```
import telepot
import subprocess
import urllib3
import time
import cv2
import torch
from models.experimental import attempt_load
from utils.general import non_max_suppression, scale_coords
from utils.datasets import letterbox
from utils.plots import plot_one_box


# Se carga el modelo entrenado con YOLO
modelo = attempt_load('/home/juan/Downloads/Modelos YOLO entrenados/modelo_yolov7_final.pt', map_location='cpu')


# Configurar el modelo para la evaluación
modelo.eval()
MAX_RETRIES = 3  # Numero máximo de intentos para reconectarse con la API de Telegram
bot = telepot.Bot('6962981081:AAHcsX22wCbqySMlJqsOKhCY7LK6PLzK-NI')
def handle(msg):
```

```
    content_type, chat_type, chat_id = telepot.glance(msg)
    if content_type == 'text':

        # Si el mensaje es el comando /foto se ejecuta el sistema
        if msg['text'] == '/foto':
            # Captura una imagen con libcamera-still y la guarda en un archivo temporal
            subprocess.run(["libcamera-still", "-o", "/home/juan/Downloads/deteccion.jpg"])
            # Cargar la imagen
            imagen = cv2.imread('/home/juan/Downloads/imagenes de prueba/61.png')
            imagen_original = imagen.copy()  # Guardar una copia de la imagen original

            # Preprocesar la imagen
            imagen = letterbox(imagen, new_shape=(640, 640))[0]
            # Convertir la imagen a tensor y normalizarla
            tensor_imagen = torch.from_numpy(imagen).float() / 255.0
            tensor_imagen = tensor_imagen.permute(2, 0, 1).unsqueeze(0)

            # Realizar la detección de personas
            with torch.no_grad():
                pred = modelo(tensor_imagen, augment=False)[0]

            # Aplicar la supresión no máxima
            pred = non_max_suppression(pred, 0.5, 0.45, classes=0, agnostic=False)

            # Dibujar los cuadros delimitadores y las etiquetas
            num_personas = 0
            for i, det in enumerate(pred):
                if len(det):
                    det[:, :4] = scale_coords(tensor_imagen.shape[2:], det[:, :4],
imagen_original.shape).round()
```

```
                for *xyxy, conf, cls in reversed(det):
                    label = 'persona'
                    plot_one_box(xyxy, imagen_original, label=label, color=(255, 0, 0),
line_thickness=2)
                    num_personas += 1

        # Guardar la imagen
        cv2.imwrite('/home/juan/Downloads/deteccion.jpg', imagen_original)
        # Imprimir el número de personas detectadas
        #print(f'Se detectaron {num_personas} {label} {"s" * (num_personas > 1)}.')
        # Lee la foto tomada con la cámara y envía la foto al chat
        with open('/home/juan/Downloads/deteccion.jpg', 'rb') as photo:
            bot.sendPhoto(chat_id, photo)

        if num_personas == 1:
            enviar_mensaje_con_reintentos(chat_id, f"*****ALERTA***** \n Se
detectó 1 persona")
        elif num_personas > 1:
            enviar_mensaje_con_reintentos(chat_id, f"*****ALERTA***** \n Se
detectaron {num_personas} {label}{'s' * (num_personas > 1)}")
        elif num_personas == 0:
            enviar_mensaje_con_reintentos(chat_id, f"No se han encontrado personas")

# Función para restablecer la conexión con la API debido a la inactividad
def enviar_mensaje_con_reintentos(chat_id, mensaje, intentos=MAX_RETRIES):
    while intentos > 0:
        try:

            bot.sendMessage(chat_id, mensaje)
            break  # Si la operación es exitosa, salir del bucle de reintentos
```

```
        except urllib3.exceptions.ProtocolError as e:
            print(f"Error de conexión: {e}")
            intentos -= 1
            if intentos > 0:
                print(f"Reintentando en 5 segundos...")
                time.sleep(5)
            else:
                print("Agotados los intentos. No se pudo enviar el mensaje.")
                break
  # Indica al bot que use la función handle cuando reciba un mensaje
bot.message_loop(handle)
# Mantiene el programa en ejecución
while True:
    pass
```