TÍTULO DEL PROYECTO	Predicción de caídas de cama hospitalaria de pacientes críticos mediante visión artificial		
AUTOR	Ismael Barakat Fernández		
DIRECTOR / PONENTE	Pedro Corcuera Miró Quesada		
TITULACIÓN	Máster Universitario en Ingeniería Industrial	FECHA	SEPT 2024

PALABRAS CLAVE

Visión artificial, Inteligencia artificial, Detección de caídas.

PLANTEAMIENTO DEL PROBLEMA

En el siguiente proyecto se abordará la implementación de un sistema de detección de caídas mediante visión artificial en pacientes en camas hospitalarias. La intención es crear un sistema completo que permita tanto monitorear al paciente como predecir con suficiente tiempo de antelación la caída de estos, ya que el sistema está enfocado a pacientes con estado de semiinconsciencia.

DESCRIPCIÓN DEL PROYECTO

La intención del proyecto es crear un producto completo en todos sus puntos, desde la construcción del sistema hasta la interfaz gráfica del usuario, pasando por la detección de las situaciones peligrosas. Así, mediante la utilización de visión artificial se podrá extraer la posición del paciente, así como relación de esta con su entorno. Posteriormente, se aplicarán lógicas comunes de relación entre la posición del cuerpo y su entorno en situaciones que preceden a caídas de cama de estas características. El objetivo final es lograr una herramienta de uso para los técnicos sanitarios, que le permita asistir a dichos pacientes de forma previa a la caída. Cabe destacar que la naturaleza propia de los pacientes es de vital importancia, ya que al encontrarse en estado de semiinconsciencia sus movimientos son lentos y sin propósito fijo, por lo que permite a los sanitarios asistirlos antes del evento.

CONCLUSIONES / PRESUPUESTO

Se consigue la construcción del sistema y sus métricas consiguen adaptarse a los requerimientos que se marcaban con objetivo. El sistema se constituye como una herramienta estable y viable, que puede ser usada como asistencia ante las situaciones propuestas. El objetivo se consigue tras la utilización de tres modelos de visión artificial y la contenerización del sistema para su

estabilidad. Así, las alarmas se reportan al usuario mediante una interfaz gráfica que alerta al técnico con ostensibles señales luminosas. En este sentido, también se plantean propuestas de mejoras estructuradas en una segunda fase.

BIBLIOGRAFÍA

[1] OMS, «Plan de Acción Mundial para la Seguridad del Paciente 2021-2030».

[2] M. Boot, J. Allison, J. Maguire, y G. O'Driscoll, «QI initiative to reduce the number of inpatient falls in an acute hospital Trust», *BMJ Open Qual*, vol. 12, n.º 1, feb. 2023, doi: 10.1136/bmjoq-2022-002102.

[3] «La Junta pagará más de 45.000 euros a los hijos de una mujer que murió en una residencia tras una caída de la cama». Accedido: 20 de agosto de 2024. [En línea]. Disponible en: https://www.diariodesevilla.es/juzgado_de_guardia/actualidad/junta-pagara-45000-euros-muerte-residencia-caida_0_2002217692.html

[4] T. Health y H. Services, «Evidence-Based Best Practices: Physical Restraints».

[5] «Prevención de caídas en pacientes hospitalizados - Manuales Clínicos». Accedido: 30 de agosto de 2024. [En línea]. Disponible en: https://manualclinico.hospitaluvrocio.es/procedimientos-generales-de-enfermeria/cuidados-basicos/prevencion-de-caidas-en-pacientes-hospitalizados/

[6] «[1511.08458] An Introduction to Convolutional Neural Networks». Accedido: 30 de agosto de 2024. [En línea]. Disponible en: https://ar5iv.labs.arxiv.org/html/1511.08458

[7] M. D. Zeiler y R. Fergus, «Stochastic pooling for regularization of deep convolutional neural networks», *1st International Conference on Learning Representations, ICLR 2013 - Conference Track Proceedings*, 2013.

[8] N. Scholz, J. Moll, M. Mälzer, K. Nagovitsyn, y V. Krozer, «Random bounce algorithm: real-time image processing for the detection of bats and birds: Algorithm description with application examples from a laboratory flight tunnel and a field test at an onshore wind energy plant», *Signal Image Video Process*, vol. 10, n.º 8, pp. 1449-1456, nov. 2016, doi: 10.1007/s11760-016-0951-0.

[9] «Detección de objetos I: YOLO • Un artículo de LMO». Accedido: 30 de agosto de 2024. [En línea]. Disponible en: https://lamaquinaoraculo.com/deep-learning/deteccion-de-objetos/

[10] «YOLOv7 Pose vs MediaPipe in Human Pose Estimation». Accedido: 30 de agosto de 2024. [En línea]. Disponible en: https://learnopencv.com/yolov7-pose-vs-mediapipe-in-human-pose-estimation/

[11] «imgaug — imgaug 0.4.0 documentation». Accedido: 27 de agosto de 2024. [En línea]. Disponible en: https://imgaug.readthedocs.io/en/latest/

[12] A. Saxena *et al.*, «A review of clustering techniques and developments», *Neurocomputing*, vol. 267, pp. 664-681, dic. 2017, doi: 10.1016/j.neucom.2017.06.053.

PROJECT TITLE	Prediction of hospital bed falls in critical patients using artificial vision		
AUTHOR	Ismael Barakat Fernández		
DIRECTOR / SPEAKER	Pedro Corcuera Miró Quesada		
DEGREE	Master's Degree in Industrial Engineering	DATE	SEPT 2024

KEYWORDS

Artificial Vision, Artificial Intelligence, Fall Detection.

PROBLEM STATEMENT

This project will focus on implementing a fall detection system using artificial vision for patients in hospital beds. The goal is to create a comprehensive system that not only monitors the patient but also predicts falls with sufficient lead time, as the system is geared toward semi-conscious patients.

PROJECT DESCRIPTION

The aim of the project is to create a fully developed product, from system construction to the user interface, while also ensuring the detection of dangerous situations. Using artificial vision, the patient's position and its relationship to their surroundings can be assessed. By extracting common patterns in body position relative to the environment, the system will predict bed falls. The final goal is to provide a tool for healthcare professionals to assist these patients before a fall occurs. Due to the semi-conscious state of the patients, their slow and purposeless movements give healthcare staff the opportunity to intervene in time.

CONCLUSIONS/BUDGET

The system has been successfully built, meeting the required objectives. It stands as a stable and viable tool that can be used in the proposed scenarios. The goal was achieved using three artificial vision models and the containerization of the system to ensure stability. Alarms are reported to the user via a graphical interface with noticeable visual signals for healthcare technicians. Additionally, structured improvement proposals are presented for a second phase.

BIBLIOGRAPHY

[1] OMS, «Plan de Acción Mundial para la Seguridad del Paciente 2021-2030».

[2] M. Boot, J. Allison, J. Maguire, y G. O'Driscoll, «QI initiative to reduce the number of inpatient falls in an acute hospital Trust», *BMJ Open Qual*, vol. 12, n.º 1, feb. 2023, doi: 10.1136/bmjoq-2022-002102.

[3] «La Junta pagará más de 45.000 euros a los hijos de una mujer que murió en una residencia tras una caída de la cama». Accedido: 20 de agosto de 2024. [En línea]. Disponible en: https://www.diariodesevilla.es/juzgado_de_guardia/actualidad/junta-pagara-45000-euros-muerte-residencia-caida_0_2002217692.html

[4] T. Health y H. Services, «Evidence-Based Best Practices: Physical Restraints».

[5] «Prevención de caídas en pacientes hospitalizados - Manuales Clínicos». Accedido: 30 de agosto de 2024. [En línea]. Disponible en: https://manualclinico.hospitaluvrocio.es/procedimientos-generales-de-enfermeria/cuidados-basicos/prevencion-de-caidas-en-pacientes-hospitalizados/

[6] «[1511.08458] An Introduction to Convolutional Neural Networks». Accedido: 30 de agosto de 2024. [En línea]. Disponible en: https://ar5iv.labs.arxiv.org/html/1511.08458

[7] M. D. Zeiler y R. Fergus, «Stochastic pooling for regularization of deep convolutional neural networks», *1st International Conference on Learning Representations, ICLR 2013 - Conference Track Proceedings*, 2013.

[8] N. Scholz, J. Moll, M. Mälzer, K. Nagovitsyn, y V. Krozer, «Random bounce algorithm: real-time image processing for the detection of bats and birds: Algorithm description with application examples from a laboratory flight tunnel and a field test at an onshore wind energy plant», *Signal Image Video Process*, vol. 10, n.º 8, pp. 1449-1456, nov. 2016, doi: 10.1007/s11760-016-0951-0.

[9] «Detección de objetos I: YOLO • Un artículo de LMO». Accedido: 30 de agosto de 2024. [En línea]. Disponible en: https://lamaquinaoraculo.com/deep-learning/deteccion-de-objetos/

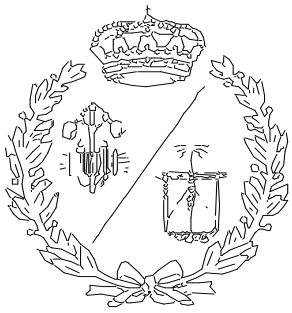
[10] «YOLOv7 Pose vs MediaPipe in Human Pose Estimation». Accedido: 30 de agosto de 2024. [En línea]. Disponible en: https://learnopencv.com/yolov7-pose-vs-mediapipe-in-human-pose-estimation/

[11] «imgaug — imgaug 0.4.0 documentation». Accedido: 27 de agosto de 2024. [En línea]. Disponible en: https://imgaug.readthedocs.io/en/latest/

[12] A. Saxena *et al.*, «A review of clustering techniques and developments», *Neurocomputing*, vol. 267, pp. 664-681, dic. 2017, doi: 10.1016/j.neucom.2017.06.053.

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Proyecto Fin de Máster

PREDICCIÓN DE CAÍDAS DE CAMA HOSPITALARIA DE PACIENTES CRÍTICOS MEDIANTE VISIÓN ARTIFICIAL

(Prediction of hospital bed falls in critical patients using artificial vision)

Para acceder al Título de

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

Autor: Ismael Barakat Fernández

Septiembre – 2024

RESUMEN

En el siguiente proyecto se abordará la implementación de un sistema de detección de caídas mediante visión artificial de pacientes en camas hospitalarias. Este tipo de situaciones generan graves complicaciones para los pacientes, y pueden ser fácilmente evitables si los profesionales sanitarios disponen de herramientas de aviso apropiadas.

El sistema usará algoritmos de inteligencia artificial orientados a la visión artificial para detectar elementos clave, en tres modelos diferentes. Un modelo será capaz de distinguir entre el paciente y otras personas en la sala. Un segundo modelo será capaz de detectar los puntos clave del cuerpo del paciente. El tercer modelo será capaz de determinar la posición relativa de esos puntos.

Posteriormente, se realizará el desarrollo y el despliegue del sistema, que podrá dar a información y señales de aviso al personal sanitario con el fin de que puedan impedir las caídas antes de que ocurran. El sistema se basará en una lógica que relacione patrones de movimientos característicos de situaciones previas a caídas, apoyándose en los puntos clave del cuerpo del paciente. Los técnicos sanitarios dispondrán de una interfaz de usuario donde podrán directamente monitorear al paciente, la cual hará los avisos correspondientes.

Se abordarán todos los aspectos claves relacionados con la planificación, ordenación, presupuesto y documentación necesaria para la realización del proyecto.

Palabras clave: Visión artificial, Inteligencia artificial, Detección de caídas.

ABSTRACT

The following project will focus on the implementation of a fall detection system using computer vision for patients in hospital beds. These situations can lead to serious complications for patients and can be easily avoided if healthcare professionals have appropriate alert tools.

The system will use artificial intelligence algorithms geared towards computer vision to detect key elements in three different models. One model will be capable of distinguishing between the patient and other people in the room. A second model will detect the key points of the patient's body. The third model will determine the relative position of these points.

Subsequently, the system will be developed and deployed, providing information and alert signals to healthcare personnel to prevent falls before they occur. The system will be based on logic that relates movement patterns characteristic of situations preceding falls, relying on the key points of the patient's body. Healthcare technicians will have access to a user interface where they can directly monitor all patients, with the appropriate alerts being issued.

All key aspects related to planning, organization, budgeting, and necessary documentation for the project will be addressed.

Keywords: Computer vision, Fall detection system, Artificial intelligence.

A mis padres.

A mis compañeros de Siali. En especial a Abel, por haberme enseñado portugués, y a Daniel, por enseñarme a buscar lo que no funciona.

CONTENIDO

Índice de figuras

1 Memoria

1.1 Introducción

El mundo de la inteligencia artificial y la automatización de procesos industriales han ido de la mano desde la llegada de esta tecnología. En este sentido, se pretende dar un nuevo enfoque a esta metodología de trabajo, ampliando la zona de actuación posible y llevándola a nuevas aplicaciones empresariales. La seguridad de los pacientes es un eje fundamental para las empresas sanitarias, donde las aplicaciones de la inteligencia artificial y de la visión artificial en particular, pueden abrir horizontes para garantizar la seguridad del paciente dejando metodologías obsoletas ética y funcionalmente.

1.1.1 Objeto

El objeto del siguiente proyecto consiste en conseguir un sistema de alertas ante caídas de pacientes de sus camas hospitalarias. Para ello, utilizaremos visión artificial, cuyo cometido será identificar la posición relativa del paciente, para posteriormente poder configurar patrones de comportamiento.

El sistema se orienta principalmente a pacientes en estado crítico, cuyo rango de movimiento es limitado. Estos pacientes habitualmente se encuentran sedados, que pueden llegar a moverse voluntariamente debido a una bajada en los fármacos sedantes, o en estado de semiinconsciencia. Este tipo de pacientes están sujetos a un gran control y seguimiento, a múltiples tratamientos y demás casuísticas que conducen a que su estabilidad sistémica sea mínima. Las consecuencias de una caída suelen ser bastante graves, incluso mortales.

Es importante remarcar que el objetivo del sistema no es reportar caídas, ya que en ese punto ya se habrían producido los daños sistémicos que se quieren evitar. Lo fundamental es impedir las lesiones que ocasione un suceso de estas características, especialmente las internas, que pueden tener consecuencias más graves.

Así, los patrones de movimientos que conducen a una caída son a velocidades muy reducidas. Esta dinámica de movimientos del paciente y su velocidad permiten tener tiempo suficiente para dar señales de alerta al personal presente y evitar la caída.

1.1.2 Alcance

El alcance del proyecto consiste en la implementación total de un sistema que avise de forma total de cualquier caída del paciente. Para ello, el proyecto hará detecciones en la misma habitación del paciente. Estará dotada de una cámara tipo CCTV. Así, el alcance comprende el desarrollo y despliegue del software, no así lo relacionado con la instalación física de las cámaras.

Para lograr tal fin se entrenarán tres modelos de la red convolucional YOLO, para:

- Poder discernir entre paciente el paciente y otras personas de la sala.
- Localizar los puntos de interés del cuerpo del paciente.
- Poder obtener su ubicación relativa respecto a la cama.

Asimismo, se realizará la lógica completa que permita obtener las señales de alarma y todo el desarrollo posterior que posibilite construir un sistema estable con el que los técnicos sanitarios puedan interactuar.

Se incluirá una descripción completa del proyecto, que incluirá una correcta planificación del proyecto, un cronograma y un presupuesto.

1.2 Estudio del arte

1.2.1 Sector médico y preocupaciones sobre la seguridad del paciente.

1.2.1.1 Introducción.

Dentro del sector médico, tanto público como privado, existe un interés creciente en torno a la seguridad del paciente. Este aumento de la importancia de la seguridad del paciente se expresa tanto por la OMS, con su *Plan de Acción Mundial para la Seguridad del Paciente 2021-2030*, como por los organismos sanitarios a nivel nacional, tanto públicos

como privados. En este sentido, los hospitales privados quieren destacar cada vez más por ofrecer garantías en materia de seguridad para sus pacientes, y diferenciarse en este aspecto sobre otros centros.

Así, aspectos cruciales como impedir caídas en situaciones de vulnerabilidad sin restricciones físicas como cinturones y otros elementos, sometidos a una crítica cada vez mayor, constituyen una línea de trabajo interesante.

1.2.1.2 Seguridad del paciente: visión general.

La seguridad de paciente ha constituido una de las principales preocupaciones en relación con la buena gestión sanitaria. Cada año, millones de pacientes sufren daños perfectamente evitables durante tratamientos o mientras reciben atención médica. Los países con ingresos bajos y medianos son particularmente vulnerables a estas dinámicas, donde ocurren más de 130 millones de accidentes de esta índole provocando 2,6 millones de muertes anuales, afectando tanto a pacientes como a la salud psicológica de los profesionales.

El *Plan de Acción Mundial para la Seguridad del Paciente 2021-2030* está basado en siete principios fundamentales que orientan su implementación. Estos consisten en:

- Involucrar a los pacientes y a las familias como socios en la atención segura.
- Lograr resultados mediante el trabajo en colaboración.
- Analizar y compartir datos para generar conocimientos.
- Traducir la evidencia en mejoras factibles y cuantificables.
- Basar las políticas y medidas en las características del entorno de atención de salud.
- Utilizar tanto los conocimientos científicos como la experiencia del paciente para mejorar la seguridad.
- Inculcar una cultura de la seguridad en el diseño y la prestación de la atención de salud.

Dentro de estos puntos, destacan tanto el segundo como el sexto. El segundo comenta la necesidad de la colaboración multidisciplinaria entre diferentes profesionales para diseñar y aplicar medidas efectivas. El sexto apartado, relativo a la utilización de los

conocimientos científicos para mejorar la seguridad. Así, se plantea el uso de innovaciones en cualquier campo y el análisis de datos de incidentes para mejorar las prácticas a nivel internacional.

En cuanto a los objetivos estratégicos expuestos, destacan:

- Objetivo estratégico 1. Políticas para eliminar los daños evitables en la atención de salud: aborda el concepto de daño evitable nulo como una mentalidad y una norma en la prestación de la atención sanitaria. Así, se plantea diseñar políticas que promuevan abordar las situaciones de riego de manera sistemática, integrando herramientas y monitoreando de forma continua los datos.
- Objetivo estratégico 2. Sistemas de alta fiabilidad: marca como prioridad estableces sistemas seguros y eficientes que sean capaces de sobreponerse a funcionamientos imprevistos.
- Objetivo estratégico 3. Seguridad de los procesos clínicos: aborda la seguridad del proceso como piedra angular del mismo, independientemente que sea simple o complejo. Así, se debe trabajar por el desarrollo de cualquier herramienta que garantice que estos procesos pueden llevarse a cabo con la mayor fiabilidad posible.[1]

1.2.1.3 Caídas en entornos hospitalarios.

Como se ha comentado, constituye una de las principales preocupaciones en los entornos hospitalarios. Así, se estima que en un hospital que pueda tener unas 800 camas, ocurren unas 1700 caídas anuales.

Estos accidentes se traducen en múltiples inconvenientes. Las complicaciones de una caída de estas características, pese a no ser mortales, pueden ser nefastas más allá de la lesión en sí. El sobredimensionamiento de procedimientos, la saturación del personal, e incluso, el sobrecoste de los tratamientos. Las caídas afectan notablemente al ánimo y la recuperación funcional de los pacientes, eternizando su estancia hospitalaria.

Evidentemente, el sobrecoste derivado de este tipo de suceso tiende a focalizarse en la sanidad privada, aunque también afecta ampliamente al sector público. En el ámbito

privado acepta en una multitud de vertientes. Puede expresarse como indemnizaciones para pacientes, sobrecostes en procedimientos de coste cerrado y, sobre todo, desconfianza del cliente, seguros, y extendida en nuestro país, la administración pública, que también se apoya en entidades privadas bajo concesiones, y que pueden retirarse o no concederse por estos hechos.

En relación con la administración pública, estas cifras pueden ascender de forma vertiginosa. Pueden parecer casos aislados, que tienen una repercusión económica leve, pero se estima que un sistema de salud como el británico tiene un sobrecoste de 630 millones de libras relacionado con sucesos de esta índole.[2], [3]

Las caídas son resultado de una interacción compleja de las eventualidades que se dan en los hospitales, donde confluyen enfermedades agudas, cambios de entorno y otros factores que alteran la percepción del paciente. Identificar estos pacientes sin ponerles medidas efectivas que impidan los accidentes da una sensación extremadamente peligrosa de seguridad. Sin embargo, pese a los intensivos estudios realizados en estas áreas, no se ha conseguido estandarizar universalmente unas dinámicas que aseguren de forma efectiva la seguridad del paciente.

Medidas de monitorización de las caídas en hospital con el objetivo de lograr paliar estas situaciones mediante diferentes procedimientos derivados de estudios clínicos, han conseguido resultados positivos en estos ámbitos, sin embargo, los resultados están lejos de ser plenamente satisfactorios.[2]

1.2.1.4 Metodologías tradicionales para prevenir caídas.

Las metodologías tradicionales para prevenir las caídas se basaban principalmente en medidas físicas de restricción de movilidad, tales como correas. Así, estas medidas también han sido utilizadas para impedir que los pacientes interfieran con los dispositivos médicos de tratamiento.

Previo a su uso, existe una evaluación clínica del paciente que estudia su historia médica y su riesgo real, a la vez que se plantea un plan de cuidados centrado en que las medidas implementadas tengan una duración temporal.

Dentro de los dispositivos más comunes encontramos las barandillas laterales, cinturones de seguridad, mesas de bandeja, cojines, etc. Asimismo, entrarían dentro de este apartado práctico como colocar la cama contra paredes o utilizar alamas disuasorias al movimiento.

Pese a ser agresivas visual y éticamente para la comunidad, siempre han sido avaladas por su efectividad. Pero la evidencia actual cuestiona su efectividad real y añade otro debate sobre si estas prácticas generan efectos adversos, incluyendo lesiones de alta gravedad si se produce una caída.

Añadidas a este peligro, se encuentran las lesiones derivadas de su uso, como pueden ser las úlceras, el desacondicionamiento físico y la incontinencia, aunque pueden ser fáciles de rehabilitar. Pero existe una preocupación mayor por los efectos psicológicos de estas prácticas, que pueden incluir ira, frustración o miedo, que disminuyen la calidad de vida del paciente.

Sin embargo, de forma tradicional, se justifica el uso de estas medidas como último recurso bajo justificación médica clara con el fin del proteger al paciente o al equipo médico. Sin embargo, una reevaluación continua del paciente siempre está indiciada, buscando aquellas medidas que puedan ofrecer los mejores resultados siendo lo menos restrictivas posible.[4]

1.2.1.5 Limitaciones y defectos de las metodologías tradicionales.

Como se ha comentado con anterioridad, existe una falta de evidencia sólida en este campo y es asociada a factores de riesgo altos. Además, estos métodos fallan en su objetivo de valorar de una forma medida los riesgos, y caen en el uso de medidas estandarizadas para todos los pacientes.

Esta valoración errónea viene dada por el uso estandarizado de evaluadores, como cuestionarios generales, y llevan aparejadas en muchos casos evaluaciones, no estandarizadas, sino directamente incorrectas. La falta de personalización de los tratamientos merma de forma notable su efectividad

Además, muchas de estas soluciones requieren de medidas proactivas, o mínimamente proactivas por parte del paciente, en vez de medias reactivas tomadas directamente por los sistemas de prevención.[5]

1.2.1.6 Avances recientes y futuras direcciones.

En los últimos tiempos se incentiva otras técnicas menos invasivas como pueden ser la implementación de programas de ejercicio, el uso de dispositivos de asistencia apropiados, modificaciones ambientales o cambios lumínicos. Todo ello se enmarca en una metodología que busca un enfoque proactivo para reducir la dependencia de barreras físicas. Además, añadido a esto se recomienda en monitoreo continuo para garantizar la efectividad y la seguridad. [4]

1.2.2 Deep Learning y redes convolucionales.

El auge de las redes neuronales artificiales en los últimos tiempos ha sido una revolución en cuanto a aprendizaje en el mundo de las inteligencias artificiales, donde destacan las redes neuronales convolucionales, las cuales se utilizan principalmente para resolver tareas difíciles de reconocimiento de patrones basadas en imágenes.

Las RNAs son sistemas de procesamiento computacional que se inspiran en el funcionamiento biológico del cerebro humano, componiéndose de una gran cantidad de nodos computacionales interconectados. Se puede observar de forma simple el comportamiento de un RNA en la Figura 1, donde la entrada corresponde a un vector multidimensional, que va a distribuir a las capas intermedias, las cuales harán decisiones teniendo en consideración la capa anterior y sopesando si estos cambios perjudican o mejoran la salida final. En las RNA, cada perceptrón de una capa se encuentra conectado a todos los perceptrones de la siguiente capa. Tener una gran red de capas intermedias apiladas ocultas es lo que se conoce comúnmente como Deep Learning.

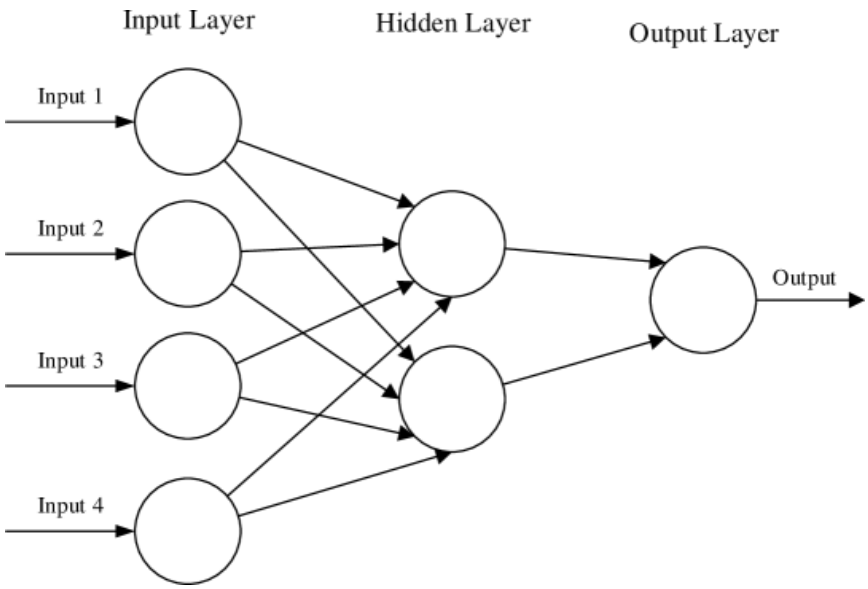


Figura 1. Estructura simplificada de un RNA.[6]

En este punto es en el que se hace una distinción entre aprendizaje supervisado y no supervisado. El aprendizaje supervisado se basa en el entrenamiento mediante entradas preetiquetadas, que actúan como dianas de aprendizaje. Este tipo de entrenamiento, que se realizará en el proyecto, busca reducir errores y ser más directo al hallar una solución, aunque puede ser más laborioso. El aprendizaje no supervisado no incluye etiqueta, y el éxito del entrenamiento se basa en sí la red reduce la función de coste asociada.

Las CNNs son análogas a las RNAs, en el sentido en el que se conforman por 'neuronas' (perceptrones) que se autooptimizan mediante el aprendizaje. Tendrá una entrada, y como resultado una función, que se denominan pesos. La diferencia práctica entre ellas reside en que las CNN son usadas para visión, ya que las RNA tienden a tener dificultades con el procesamiento de imágenes.

En cuanto a la estructura de las CNN, está basada en su función, el procesamiento de imágenes, y cómo manejar estos datos de la forma más eficiente posible. Una de las principales diferencias es que las neuronas de una CNN es que se encuentran organizadas en tres dimensiones, siendo a la tercera dimensión no referente a una capa numérica, sino a una dimensión del volumen de activación. Asimismo, las CNN solo están conectadas localmente a un subconjunto local (campo receptivo), lo que mejora la eficiencia y reduce la complejidad.

Las CNN se componen de tres tipos de capas: las convolucionales, las de agrupamiento y las completamente conectadas, que cuando se apilan, se crea una red neuronal convolucional, tal y como se puede observar en la Figura 2.

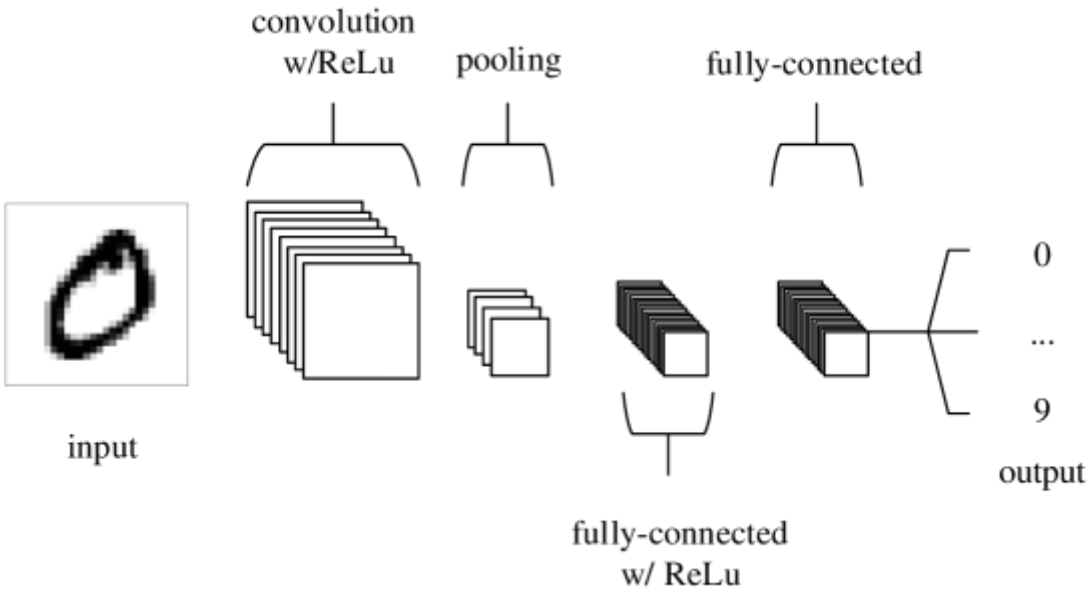


Figura 2. Estructura simplificada de una CNN.[6]

En la Figura 2 se pueden observar cuatro áreas clave:

- Capa de entrada que contiene los valores de los píxeles de imagen.
- Capa convolucional que determinará la salida de las neuronas, las cuales están conectadas a regiones locales, y cuyo cálculo sea el producto escalar entre sus pesos y la región de input que le corresponda. La unidad lineal rectificada, (ReLu), tiene como objetivo aplicar una función de activación elemento por elemento a la salida de la activación producida por la capa anterior.
- La capa de agrupamiento, que hará un muestreo y reducirá la cantidad de parámetros dentro de la activación.
- Las capas completamente conectadas, que harán las funciones tradicionales y en las que se recomienda la utilización de ReLu para mejorar el rendimiento.

Todo este funcionamiento orbita alrededor de los kernel, que habitualmente tienen una dimensión espacial reducida, pero se extiende a lo largo de toda la entrada. Cuando los datos llegan a la entrada, la capa convoluciona cada filtro y produce un mapa de activación.

A medida que se avanza por la entrada, se calcula el producto escalar de cada kernel. Así, la red aprenderá a asociar ciertos kernels a ciertas características concretas en una posición espacial determinada, lo que se conoce como activadores. Se ve en la Figura 3. Cada kernel tendrá un mapa diferente, que se apilará a lo largo de la dimensión de profundidad y dará resultado a una salida diferente.

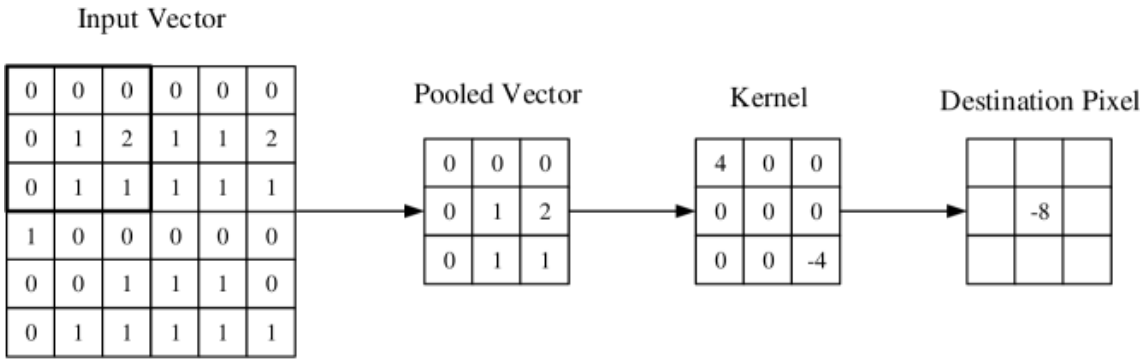


Figura 3. Representación visual de una capa convolucional.[6]

Esta capa es la que representa una optimización respecto a las RNAs, ya que reduce la complejidad del proceso y optimiza la salida. Esto se realiza a través de tres hiperparámetros, como son: la profundidad, el paso y la configuración de relleno con ceros.

La profundidad está relacionada con la conexión entre neuronas. En las RNA, el perceptrón está conectada a todas las de la siguiente capa. En las CNN, se puede reducir el número de conexiones y de neuronas. La zancada está relacionada con la profundidad dimensional. Aumentarla reduce las superposiciones. El relleno de ceros se basa en rellenar el borde de la entrada para brindar mayor control en cuanto a la dimensionalidad de la salida.

La capa de agrupamiento pretende reducir gradualmente la dimensionalidad de la representación, reduciéndolas en un 25%. Evidente, esto representa un procedimiento destructivo y solo se suelen usar dos metodologías. Se suele utilizar la agrupación superpuesta, con una zancada en 2 y un kernel de 3. Un kernel mayor de 3 desembocaría en una pérdida de rendimiento del modelo.

En la capa completamente conectada contiene neuronas conectadas como las RNAs.

Aunque es difícil definir una arquitectura fija de este tipo de redes, siguen similitudes en su estructura, poniendo en serie capas convolucionales, las de agrupamiento y las conectadas. Existe otra metodología que consiste en poner dos conjuntos de capas convolucionales delante de cada conjunto de agrupamiento. Da un mayor rendimiento en modelos con características complejas. Se observa en la Figura 4.[7], [8]

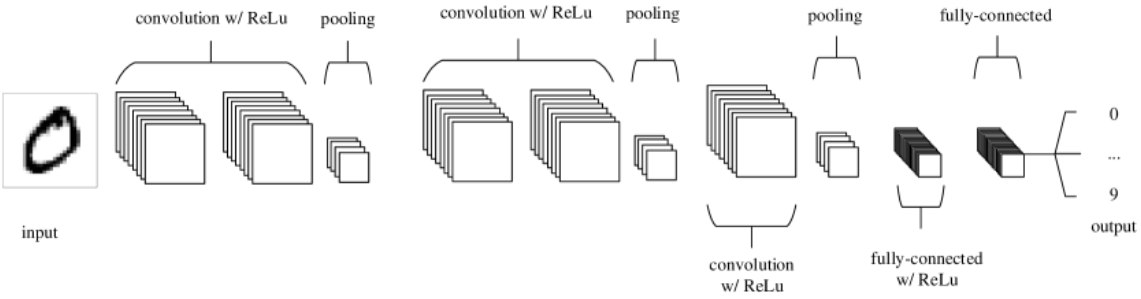


Figura 4. Configuración de intercalado de capas.[6]

1.2.2.1 Detección de Objetos mediante Visión Artificial y YOLO.

Dentro de la visión artificial, la detección de objetos constituye en campo más importante en relación con el campo de trabajo industrial. Así, mediante esta técnica se pueden localizar objetos pertenecientes a diferentes clases previamente definidas. Como particularidad, las redes diseñadas con este fin, siendo CNNs, tienen una extensa capa final completamente conectada, que es la que realiza la detección propiamente dicha.

Estas redes pueden realizar el reconocimiento de un solo paso, como es el caso de YOLO (You Only Look Once) o mediante un procedimiento en dos pasos, donde la red crea regiones candidatas que se hacen pasar por una red clarificadora puntuándolas. Si el umbral de puntuación es suficiente se considera la detección. Es propia de redes como R-CNN.

YOLO procesa la imagen dividiéndola en una cuadrícula de secciones SxS, analizando convolucionalmente cada casilla por separado. Posteriormente, cinco neuronas

predictoras de salida, con información geométrica, que predicen la detección, indican su centro, las dimensiones de la detección y la fiabilidad, en valores normalizados. La fiabilidad es el producto de la probabilidad de que existe en el objeto por el índice de Jacquard de dicha casilla, donde A es la ventana predicha y B la subyacente. Si esta fiabilidad supera el 0,5 se considera la detección del objeto. Para evitar solapamientos utilizan un algoritmo de supresión no – maximal.

$$f = P(o)\, J\,(A, B)$$

Además, cada casilla contiene neuronas localizadoras con información semántica, que determina de qué clase es la detección. Arrojan la probabilidad para el cálculo de la fiabilidad.

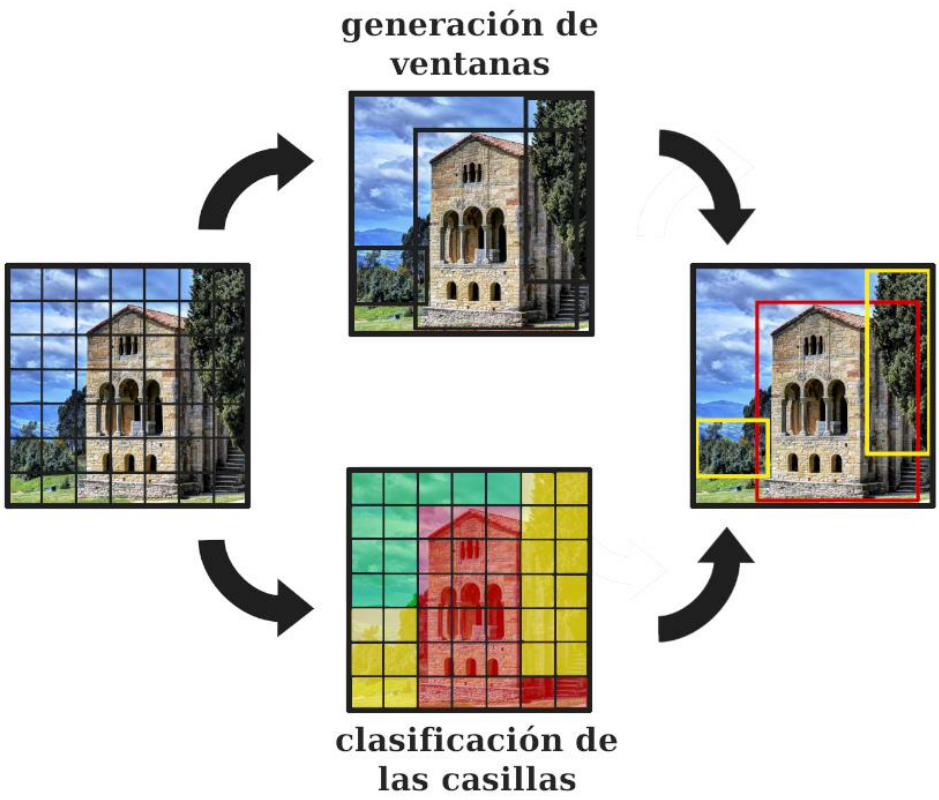


Figura 5. Detección geométrica de YOLO.[9]

El algoritmo de YOLO fijó una dimensión de 7x7, dos predictoras y 20 clasificadoras por casilla, con un total de 30 neuronas en la capa de salida.

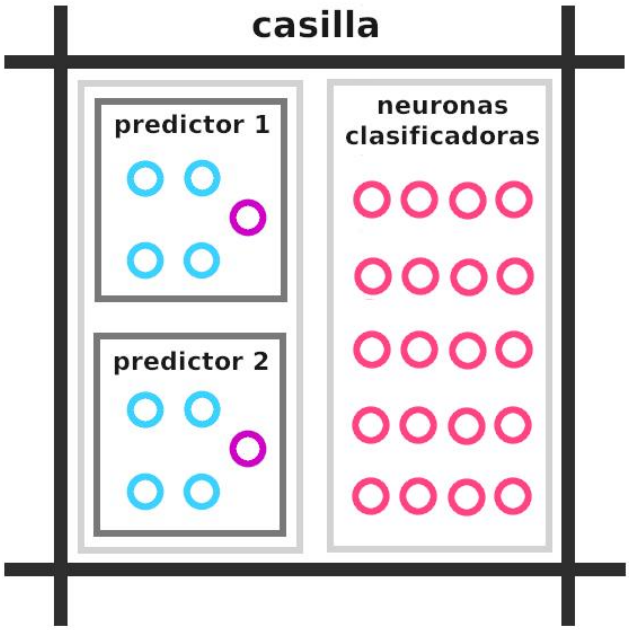


Figura 6. Configuración neuronal de una casilla.[9]

La red contiene 24 capas convolucionales intercaladas por capas de agrupación, con dos últimas densamente conectadas, teniendo las convoluciones y las de agrupamiento leaky ReLu, mientas que la última capa conectada, que tiene la función de activación ReLu tradicional.

Su función de pérdida es de tipo L2, convirtiendo el problema en un problema de regresión cuadrática. A cada ventana subyacente se le asigna un predictor responsable, y en cada uno de ellos se realiza el cálculo de la función de pérdida, pero solo harán una contribución a la función las neuronas selectoras, mientras que el resto no contribuye.

$$L_p = \lambda_c * \left[(x - x_0)^2 + (y - y_0)^2 + (b - b_0)^2 + \left(\sqrt{h} - \sqrt{h_0}\right)^2\right] + (f - f_0)^2$$

Donde λ_c está fijada en 5 y f y f_0 representan la fiabilidad esperada y real. Para que el gradiente de neuronas no crezca demasiado y conduzca a una estabilidad, las ventanas sin objetos contribuyen a la función de pérdida con L_s y las neuronas clasificadoras con L_c.

$$L_s = \lambda_s(f - f_0)^2$$

Con λ_s igual a 0.5.

$$L_c = \sum\left(p(c) - p_o(c)\right)^2$$

Siendo $p(c)$ la probabilidad de que en la casilla presente haya un objeto de clase c y $p_o(c)$ es la fracción de casilla que en la subyacente está cubierta por el objeto tipo c.

Así, finalmente la función de pérdida queda expresada como el sumatorio de las anteriores. El gradiente de esta función de pérdida se retropropaga a las capas iniciales. [9]

$$L = \sum L_p + \sum L_s + \sum L_c$$

1.3 Modelos

1.3.1 YOLO Pose y Keypoints.

1.3.1.1 Objetivo del modelo

La piedra angular del proyecto reside en la detección humana, así como en sus partes corporales. De esta forma, es necesario un modelo que sea capaz de detectar de una forma eficaz y fiable.

Ultralytics, de forma general, ofrece modelos ya entrenados de YOLO, ajustándose los más conocidos a la nomenclatura habitual del tamaño de modelos. Estos modelos constituyen un recurso open source generalista que permiten hacer detecciones de forma completa a objetos cotidianos.

Dentro de este universo de pesos preentrenados, contamos con los modelos POSE. Estos modelos, disponibles en todo el rango de tamaños, tienen como objetivo la detección de personas, con la particularidad de poder detectar los conocidos como Keypoints. Estos puntos clave ofrecen la ubicación exacta de localizaciones del cuerpo. Así, se identifican 17 Keypoints, observables de la misma forma en la Figura 7:

- Nariz
- Ojo izquierdo

- Ojo derecho
- Oído izquierdo
- Oído derecho
- Hombro izquierdo
- Hombro derecho
- Codo izquierdo
- Codo derecho
- Muñeca izquierda
- Muñeca derecha
- Cadera izquierda
- Cadera derecha
- Rodilla izquierda
- Rodilla derecha
- Tobillo izquierdo
- Tobillo derecho

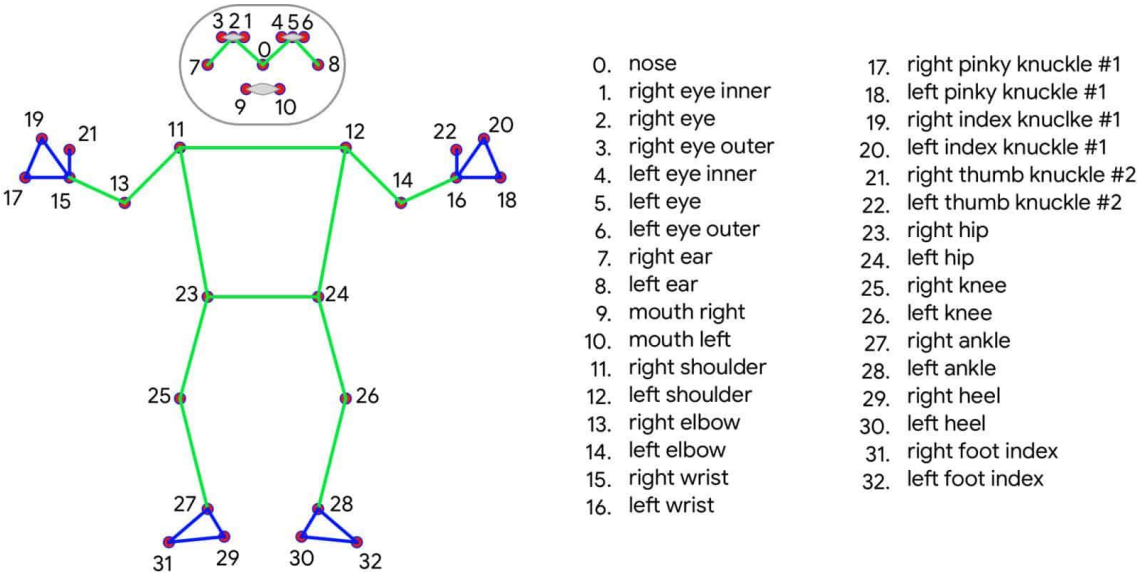


Figura 7. Posicionamiento extenso de los Keypoints.[9]

Esta detección precisa de puntos clave permite hacer lógicas de posición entre partes del cuerpo, como se puede observar en la Figura 8, donde se realizan correlaciones entre los puntos para determinar si un individuo se encuentra caído o no.[10]

Figura 8. Uso práctico de lógica de Keypoints. Elaboración Propia.

1.3.2 Modelo de Barreras de Cama.

1.3.2.1 Objetivo del modelo.

Sin duda, el posicionamiento de los Keypoints constituye el eje sobre el que se fundamente las detecciones de las caídas, y, en definitiva, la piedra angular del proyecto en sí. Sin embargo, la posición espacial de los puntos clave sin un contexto respecto al

entorno no es suficiente para determinar en este caso el resultado final del análisis. En otras situaciones, referenciadas con anterioridad, la simple posición relativa entre Keypoints podría ser suficiente para determinar el estado del individuo, sin embargo, es necesario disponer de datos de las posiciones de las camas.

Pese a que con anterioridad se ha comentado que las barreras constituyen un elemento que la ética actual en relación con la seguridad del paciente invita a eliminar, la realidad es que la práctica totalidad de las camas a estudio cuentan con estos elementos de impedimento físico. Al final, nuestro objetivo fundamental es determinar si los Keypoints se encuentran dentro o fuera de la cama. Así, es suficiente con detectar la posición de las barreras y ver si ciertos Keypoints se encuentra próximos a una salida de esta.

En este sentido, y observando la dinámica a la cual torna la ética de la seguridad del paciente, es previsible que en un futuro no lejano las barreras no formen parte indispensable de la aparamenta de una unidad médica, por lo que esa contingencia está consideraba y se podría pasar a operar con un modelo que detectase si la cama cuenta con barreras o no, y en caso de que no disponga de ellas, simplemente detectar los bordes de la cama para determinar la posición relativa de los Keypoints.

1.3.2.2 Recolección de datos y augmentación artificial del dataset.

Dado que el mundo de la medicina no permite a un acceso a la información tan extenso como otras áreas debido a la privacidad de los pacientes, y teniendo en consideración que el proyecto no tenía acceso a poder realizar esta actividad in situ, se procedió a la recolección de imágenes de forma manual.

En estos casos de aplicación, donde se busca encontrar realizar un software generalista para poder aplicarlo en infinitas situaciones sin la necesidad de realizar modificaciones extensas, se tiende a que los datasets objeto de estudio sean amplios y que cuenten con una gran variabilidad dentro ellos, es decir, mayor de la habitual. Sin embargo, dada la casuística de la aplicación y considerando la complejidad de adquirir datos, así como que la creación de un dataset de estas características de forma profesional requeriría un equipo completo, se optó por utilizar un dataset reducido, con suficiente variabilidad, y lograr la masa de entrenamiento y el aumento de la variabilidad mediante la augmentación de este.

La augmentación consiste en el proceso de multiplicar el tamaño del dataset con el fin de obtener un mayor número de imágenes con las que realizar los entrenamientos. Una red neuronal requiere variabilidad en los datos aportados con el fin de poder obtener un aprendizaje real de los datos. En este sentido, si se entrena con una misma imagen que ha sido multiplicada sin más, el resultado final será que el modelo será perfectamente capaz de reconocer lo ocurrido en la imagen, pero nada más allá de ese punto. En el momento que haya una modificación en la imagen el modelo no funcionará. Mediante este proceso el modelo adquiere el conocimiento de forma memorizada.

La augmentación rompe esta barrera utilizando modificaciones visuales sobre las imágenes. Así, modificando parámetros de estas podemos llegar a obtener un dataset de mayor tamaño al mismo tiempo que aumentamos la variabilidad de este. Estas modificaciones abarcan tanto cambios visuales propios de la imagen, como pueden ser la saturación, el brillo o el desenfoque, como cambios físicos de las mismas, como podría ser aplicar rotaciones, escalados o volteos.

Es importante recalcar que siempre que se realizan este tipo de modificaciones en los datasets, se ha de ser consciente y consecuente en los cambios que se ejecutan. Si el objetivo del modelo es la detección de un objeto de un determinado color, cambios en la saturación confundirán al modelo durante el entrenamiento y serán perjudiciales para el mismo. Cambios físicos en proyectos de OCR (Reconocimiento Óptico de Imágenes), como podría ser aplicar un volteo vertical, pueden hacer que un 6 pase a ser un 9 o viceversa. Así, a la hora de aplicar estas técnicas hay que ser cautos y plenamente conscientes de los cambios que se van a realizar.

Así, se obtuvieron un total de 142 imágenes para su posterior etiquetado. Como se expondrá posteriormente, en un principio se tomaron 39, y, por causas que se explicarán luego, se decidió ampliarlo. Las imágenes que se obtuvieron buscaban obtener la mayor variabilidad posible y tener en consideración los casos de aplicación que se darán en la realidad.

Figura 9. Imágenes utilizadas para entrenar ya etiquetadas. Elaboración propia.

Para realizar tanto el etiquetado de datos como el entrenamiento del YOLO se utilizará el software Rely Cloud de Siali Technologies, el cual permite realizar estas acciones tanto para datasets de detección, los cuales marcan con una bounding box la detección, como de segmentación, los cuales delimitan de forma precisa la detección. Dada la casuística de este caso se opta por tomar un dataset de segmentación. Así, de manera inicial se optó por etiquetar unas 39 imágenes, en una segunda fase se amplió a 93, hasta llegar a la consecución final de recolección y etiquetación de 142.

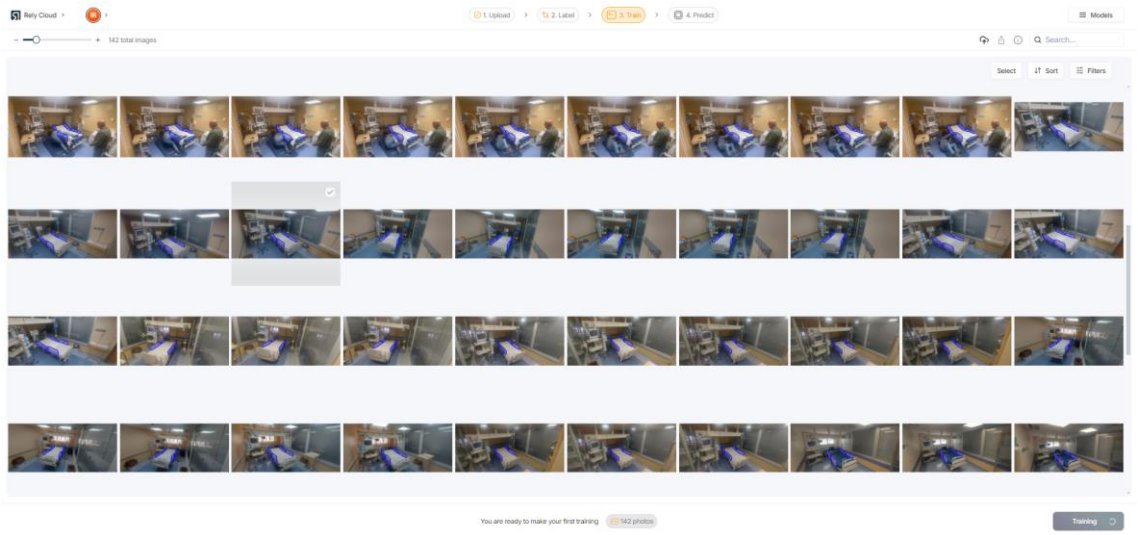


Figura 10. Herramienta de etiquetado de Rely Cloud. Elaboración propia.

Este entrenamiento inicial con 39 constituye, a simple vista para cualquier persona iniciada en las metodologías de entrenamiento, en una muestra seguramente bastante pequeña y sin la variabilidad suficiente, y especialmente si se trata de un dataset de segmentación.

La información necesaria para entrenar este dataset no es accesible de una forma sencilla para el público general. Lo relacionado con asuntos médicos está sometido a estrictas restricciones relativas a la protección de datos de los pacientes, por lo que las imágenes derivadas de estas dinámicas, incluso en internet en datasets amplios y de libre acceso suelen ser artificiales y no ajustarse a la realidad. Además, recordemos que la finalidad del proyecto es poder extraer estas imágenes de una cámara de seguridad tradicional, ubicada en una esquina de la habitación. Sumando este dato al anterior, de forma lógica los datasets generales abiertos en la red no ofrecen por norma general fotos de estas características, que serían las indicadas para el entrenamiento.

Por ello, se utiliza una técnica alternativa como es la augmentación. Para ello, se utiliza la librería de Python imgaug, la cual además está diseñada específicamente para redes convolucionales. Esta librería permite augmentar con un amplio abanico de técnicas, las cuales incluyen tanto datos relativos a la apariencia de la imagen, como podrían ser aquellos relacionados con el brillo, la saturación o el aumento de ruido, como otros relacionados con el aspecto físico de las mismas, como podrían ser giros, cambios de

perspectivas, oclusiones o reescalados, permitiendo además aplicarlos entre sí mediante métodos aleatorios de gran interés.

Sus características preferentes respectos a otras librerías de este estilo como pueden ser Albumentations o OpenCV reside de forma principal en la versatilidad y el rango de las augmentaciones. OpenCV, por ejemplo, ofrece un abanico muy amplio en transformación, modificación, guardado, visualización, etc., si bien es cierto que en el aspecto de las augmentaciones no es su punto fuerte. Además, como hemos comentado antes, ofrece particularidades especiales como la posibilidad de implementar las augmentaciones de forma estocástica y su enfoque específico en CNN. [11]

Para este dataset en particular, tanto para este como para los siguientes entrenamientos como veremos a continuación se implementaron se pueden observar en la Figura 11.

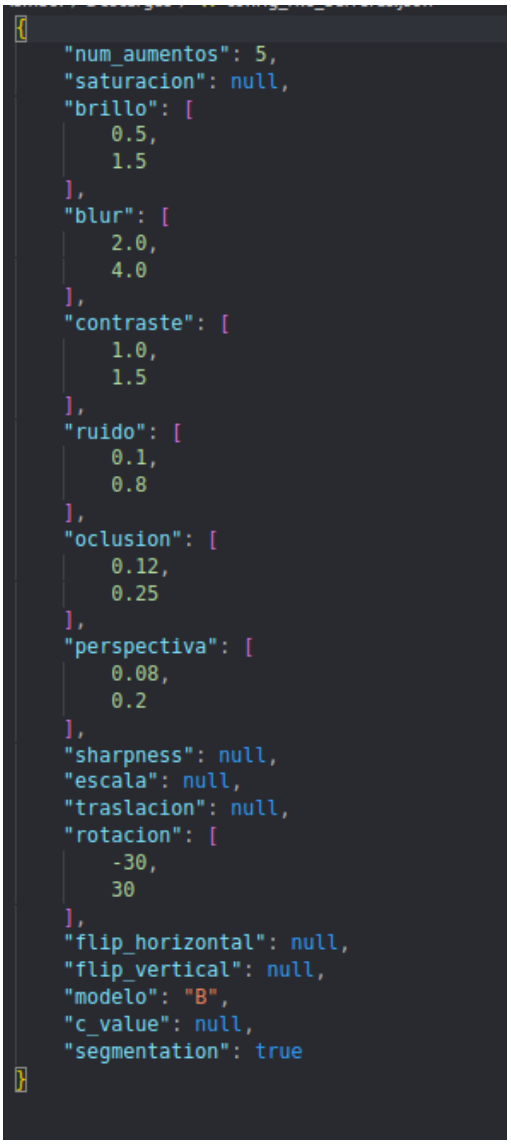


Figura 11.JSON de parámetros para augmentar. Elaboración propia.

El modelo B, indica que la augmentación de clases fue directa.

1.3.2.3 Entrenamiento y validación.

1.3.2.3.1 Configuración de entrenamientos.

En la conformación de modelos, el entrenamiento y la validación son dos conceptos consecutivos y fuertemente vinculados dentro de la visión artificial. Así, un entrenamiento es realmente exitoso cuando se encuentra validado y respaldado por datos. En este sentido, la dinámica habitual de un entrenamiento consiste en determinar los parámetros de este, que pasaremos a detallar a continuación, basados en la experiencia

del técnico en relación con la tipología de dataset que se presenta, para posteriormente una vez realizado analizar sus métricas y verificar que fue correcto.

Dentro de los modelos de IA en general, y de YOLO dentro de la visión en particular, existen una serie de parámetros configuración modificables. Estos parámetros tienen un número bastante alto, pero los que se modifican de forma general para un entrenamiento son:

1. Batch: número de imágenes que se procesan en un lote durante el entrenamiento. Debe ajustarse según la capacidad de memoria de la GPU.
2. Épocas: número total de épocas o ciclos completos sobre el conjunto de entrenamiento. Más épocas pueden conducir a un mejor ajuste, pero también al riesgo de sobreajuste.
3. Tamaño de imagen: tamaño de las imágenes de entrada. Las versiones de YOLO suelen trabajar con tamaños como 640x640 o 1280x1280. Un tamaño mayor puede mejorar la precisión a costa de un mayor uso de memoria y tiempo de entrenamiento. Así, por lo general, un modelo entrenado con un tamaño de imagen mayor trabajará peor que uno entrenado a un tamaño menor. Hay que mantener la idea del entrenamiento siempre en mente. YOLO está diseñado para poder detectar lo mismo que el ojo humano. Si lo que se quiere detectar es visible a 640, se debe entrenar a 640. El entrenamiento a mayores tamaños está reservado a detecciones pequeñas que no serían distinguibles a 640.

Los siguientes parámetros no son modificados de forma habitual, pero tienen una alta relevancia y es positivo conocerlos para tener una mayor comprensión del proceso:

1. Learning rate: tasa de aprendizaje que controla qué tan rápido se ajustan los pesos del modelo. Es común usar técnicas de ajuste automático de la tasa de aprendizaje, como el decaimiento de la tasa de aprendizaje.
2. Momentum: Parámetro que ayuda a acelerar la convergencia del optimizador. Un valor típico es 0.9.

Así, existe un último parámetro relacionado con el modelo en sí, el tamaño. Estos se ajustan a la nomenclatura habitual de N, S, M, L y XL, de forma general. La elección de

uno u otro modelo es responsabilidad del técnico, y se basa de forma general en la visión del técnico o en lo que se denomina distanciamiento espacial de variables. Se puede entender un poco mejor este concepto en la Figura 12.

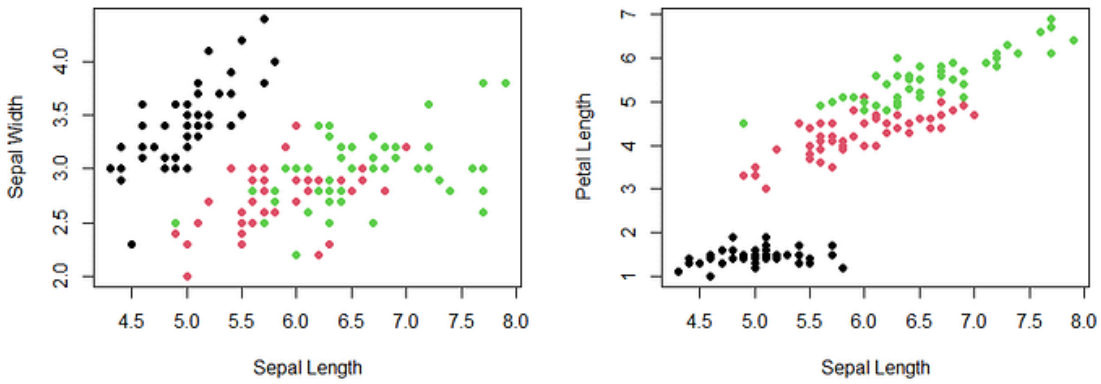


Figura 12. Representación visual del distanciamiento espacial.[12]

De una forma resumida, el distanciamiento general de variables se concibe como la diferenciación real que existe entre dos clases y la facilidad que existe para discernirlas. Así, de una forma visual, en la Figura 13,se puede asemejar aquellas gráficas con grupos absolutamente separados, como la ubicada en la primera fila de la segunda columna, con un escenario en el que existen 4 clases que son bastante diferentes unas de otras, en contraposición, por ejemplo, a la última fila, donde existe una confusión mayor entre estas a la hora de separarlas.

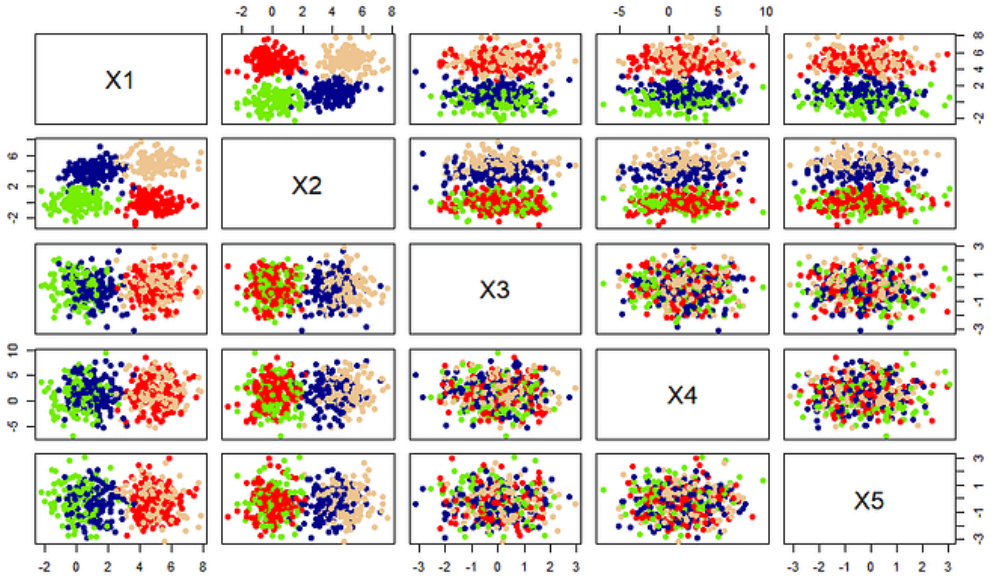


Figura 13. Ejemplos de dificultades en la diferenciación de clases.[12]

El distanciamiento espacial combinado con el número de variables determina cuál debe ser la elección del modelo. A mayor número de variables, mayor deberá ser el tamaño del modelo. Hay que ser conscientes de que un mayor modelo lleva aparejado un tiempo de inferencia mayor.

1.3.2.3.2 Primer entrenamiento.

En una primera iteración, se entrenó el modelo basado en los parámetros que se pueden observar en la Figura 14.

ARGS

aug_parameters	{"num_aumentos": 5, "saturacion": null, "brill...
aug_parameters_states	{"num_aumentos": true, "saturacion": false, "...
augmentation	True
base_model	yolov8m-seg.pt
batch_size	-1
clearML_download	True
clearML_project_name	Siali_ICUWatch_barreras
clearML_queue_name	train
clearML_task_name	barreras_aug16_7
dataset_path	/app/datasets/311/train.yaml
epochs	300
id_dataset	311
img_size	640
save_period	-1

Figura 14. Parámetros para el primer entrenamiento. Elaboración propia.

Al finalizar el entrenamiento se pasa a comprobar cómo han sido los resultados de este. Lo ideal en estos casos es hacer una doble validación, basada, por un lado, en una comprobación que podríamos denominar teórica, y además una comprobación práctica.

La comprobación práctica consiste en coger imágenes de prueba a las que el modelo no se haya enfrentado antes. En este caso, podemos realizar en un primer estado esta prueba, y veremos cuáles son las fallas del modelo, como se puede observar en la Figura 15.

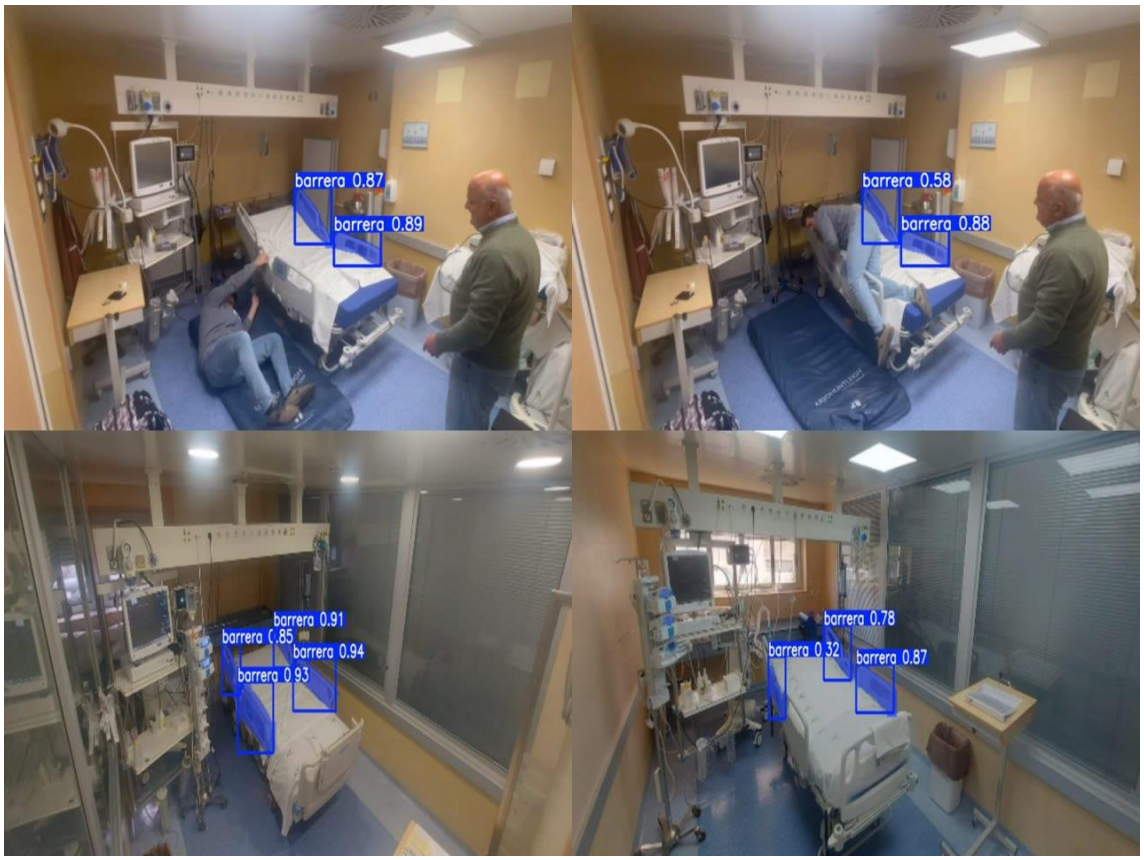


Figura 15. Pruebas prácticas de validación del primer modelo. Elaboración propia.

Sin embargo, para un técnico ya docto en la materia constituye un ejercicio relativamente rápido analizar los resultados métricos obtenidos, que no solo le aportarán al técnico información relacionada con el buen o mal funcionamiento del modelo, sino que además podrá determinar, extrayendo datos de esas métricas, a que se debe el mal funcionamiento del, modelo en caso de que se dé.

Así, existen varias métricas a tener en consideración. Por un lado, se encuentra lo que se denomina la matriz de confusión, la cual puede venir en datos absolutos (Figura 16) o en datos unitarios (Figura 17).

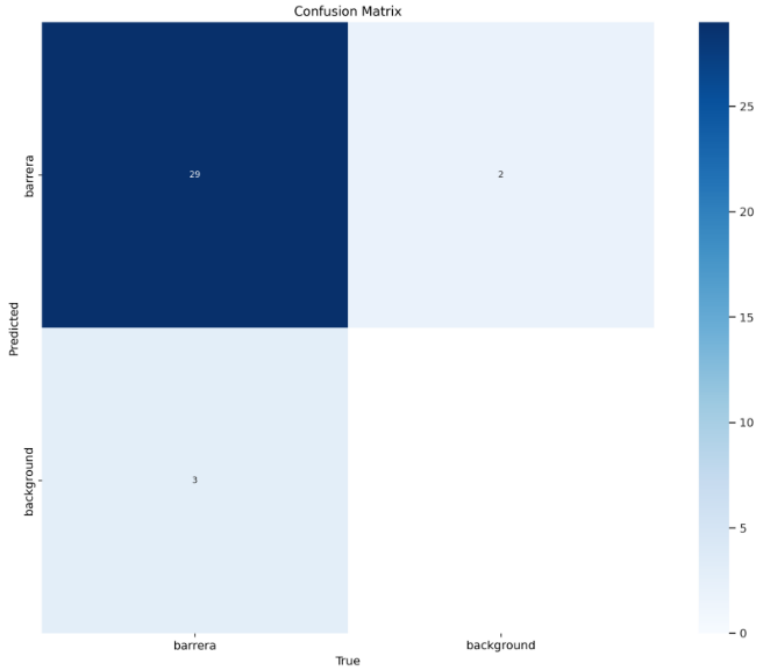


Figura 16. Matriz de confusión no normalizada del primer entrenamiento. Elaboración propia.

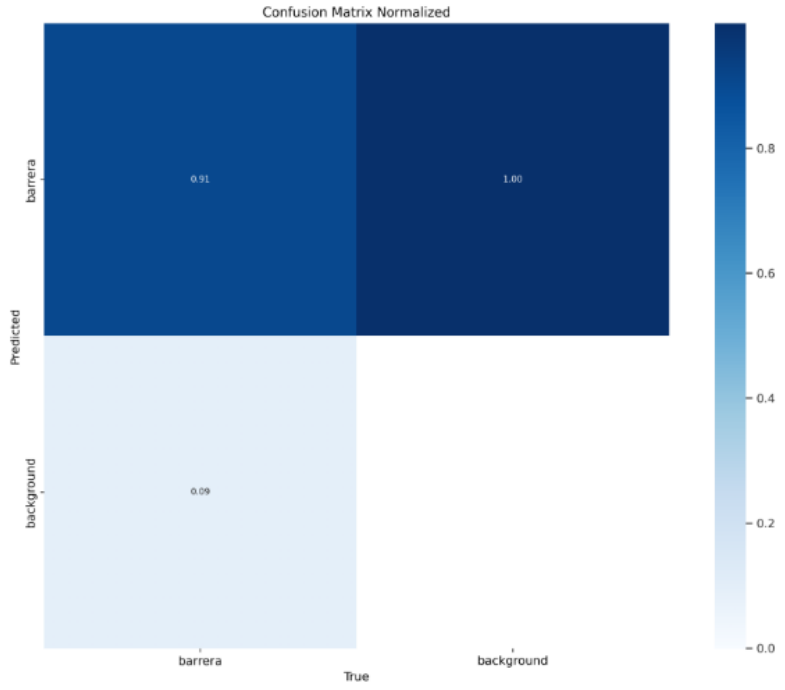


Figura 17. Matriz de confusión normalizada del primer entrenamiento. Elaboración propia.

Estas matrices muestran el total de acierto respecto a lo esperado, dentro de la validación de este. Para analizar estas métricas, es importante no perder de vista la naturaleza del dataset. En este sentido, se debe recordar que esto es un dataset de segmentación con una

única clase. La dificultad de este reside en delimitar correctamente en límite de las barreras, por lo que detectarlas debería ser algo banal. Que haya fallos en las detecciones ya es de por sí un mal indicador.

Dado que es un dataset de segmentación, el análisis de datos puede ser un poco complejo porque se multiplican las variables. Así, obtenemos las métricas representadas en la Figura 18. Cabe destacar que el análisis solo es relevante en las métricas de validación, ya que las métricas de entrenamiento siempre presentarán un recorrido positivo mejorando en cada época.

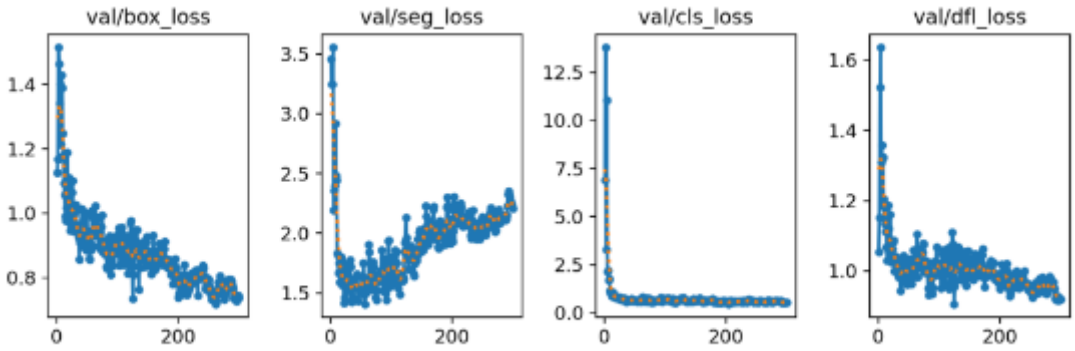


Figura 18. Métricas de pérdidas del primer entrenamiento. Elaboración propia.

Las métricas clave que analizamos son la pérdida del cuadro delimitador (box loss), la pérdida de clasificación (class loss), la pérdida relacionada con el área segmentada (seg los) y la pérdida de distribución de etiquetas (dfl loss). Estas métricas nos proporcionan información crítica sobre la precisión en la localización de objetos, la exactitud en la clasificación de estos y la optimización de la distribución de etiquetas, respectivamente. En este aspecto para el proyecto es crítica que la segmentación sea correcta.

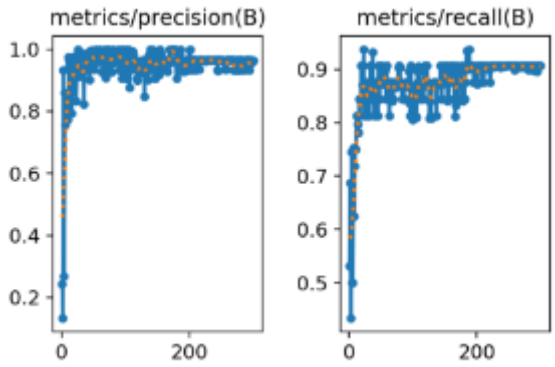


Figura 19. Métricas de precisión y recall del primer entrenamiento. Elaboración propia.

Las métricas de precisión y recall de la Figura 19 proporcionan información crucial sobre la capacidad del modelo para hacer predicciones correctas y detectar la mayoría de las instancias verdaderas de las clases de objetos en las imágenes. La precisión se refiere a la proporción de verdaderos positivos (predicciones correctas) entre todas las predicciones realizadas (verdaderos positivos y falsos positivos), indicando la exactitud del modelo cuando hace una predicción. El recall, por otro lado, mide la proporción de verdaderos positivos entre todas las instancias que deberían haber sido identificadas (verdaderos positivos y falsos negativos), reflejando la capacidad del modelo para capturar todas las instancias relevantes. En este caso, como solo se cuenta con una clase, las métricas no son malas, aunque se esperaría que estuvieran ambas por encima del 0,95.

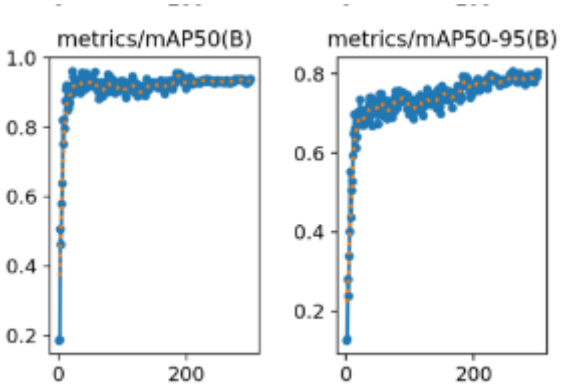


Figura 20. Métrica de mAP del primer entrenamiento. Elaboración propia.

El promedio de precisión (mean Average Precision, mAP), en la Figura 20, se calcula como el promedio de las precisiones a diferentes niveles de recall, ofreciendo una visión integral de la capacidad del modelo para detectar y clasificar objetos correctamente.

Específicamente, mAP50 evalúa el rendimiento del modelo a un umbral de intersección sobre unión (IoU) del 50%, mientras que mAP50-95 proporciona una evaluación más rigurosa al considerar múltiples umbrales de IoU, desde el 50% hasta el 95%. Estas métricas son cruciales porque reflejan tanto la precisión como la capacidad de recuperación del modelo a lo largo de un rango de condiciones, proporcionando una medida completa de su eficacia y robustez en la detección de objetos. De igual modo, ambas métricas deberían tener valores mejores en todos los escenarios propuestos, ya que solo debe identificar una clase

Así, de las pruebas prácticas y teóricas de este entrenamiento extraemos que es necesario una mayor variabilidad. Se ve que se necesita una muestra de mayor tamaño, ya que el error inicial de las estimaciones es notable, además de las fluctuaciones constantes de las gráficas. Hay demasiada inestabilidad y el aprendizaje del modelo no está siendo del todo correcto. De igual forma, estas fluctuaciones en las gráficas de modelos de segmentación son habituales, ya que el aprendizaje es más complicado. Además, notamos que el número de épocas podría llegar a ser un problema al no observar una plena convergencia.

Por pura curiosidad científica, se decide realizar el mismo entrenamiento, pero sin augmentado, los resultados, en la Figura 21, nos muestran como los errores son ostensiblemente mayores, duplicando en algunos casos, y las gráficas de fiabilidad se reducen notablemente.

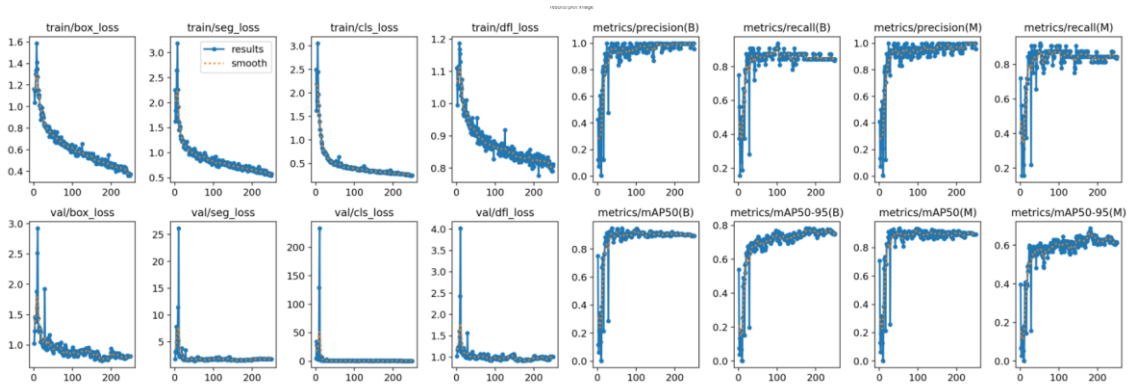


Figura 21. Métricas del primer entrenamiento sin augmentado. Elaboración propia.

1.3.2.3.3 Segundo entrenamiento.

Para afrontar un segundo entrenamiento, se decide aumentar el número de imágenes del dataset. El primer entrenamiento tenía una muestra pequeña acorde a la cantidad de imágenes que se había reunido hasta entonces. Así, en este entrenamiento se consigue reunir en torno a unas 100 imágenes, que augmentando por cinco el dataset, nos arroja ya una buena muestra de augmentado. Los parámetros funcionales del segundo entrenamiento son idénticos al primero.

Las gráficas obtenidas para este entrenamiento las podemos ver en la Figura 22, Figura 23 y Figura 24.

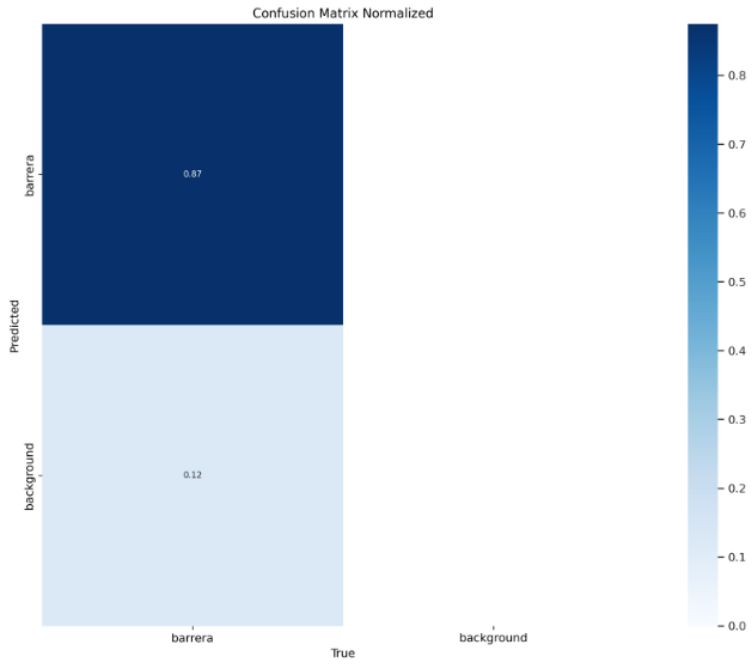


Figura 22. Matriz normalizada del segundo entrenamiento. Elaboración propia.

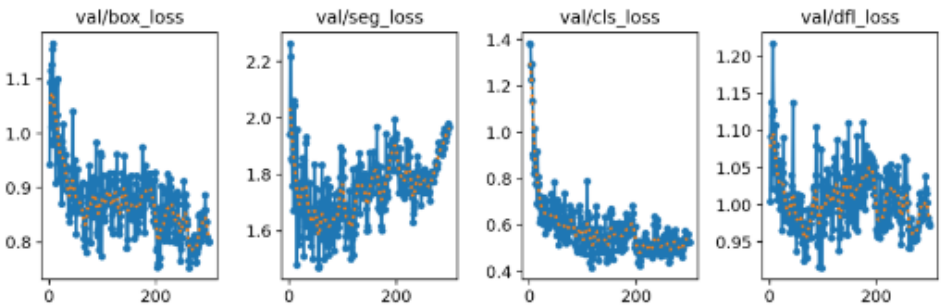


Figura 23. Métricas de pérdidas del segundo entrenamiento. Elaboración propia.

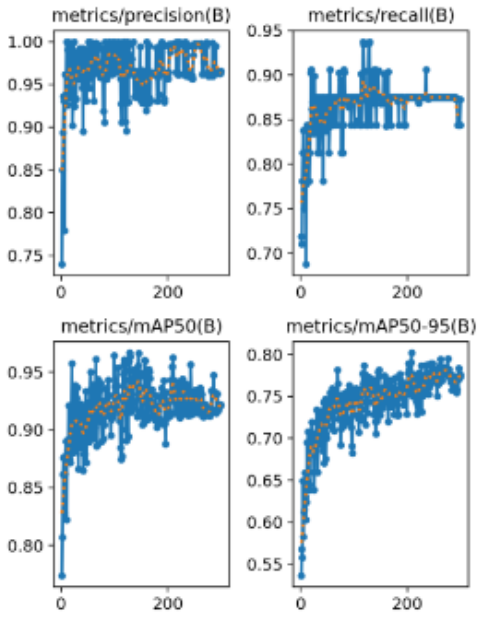


Figura 24. Métricas de precisión, recall y mAP del segundo entrenamiento. Elaboración propia.

Las pruebas prácticas de este entrenamiento se pueden ver en la Figura 25.

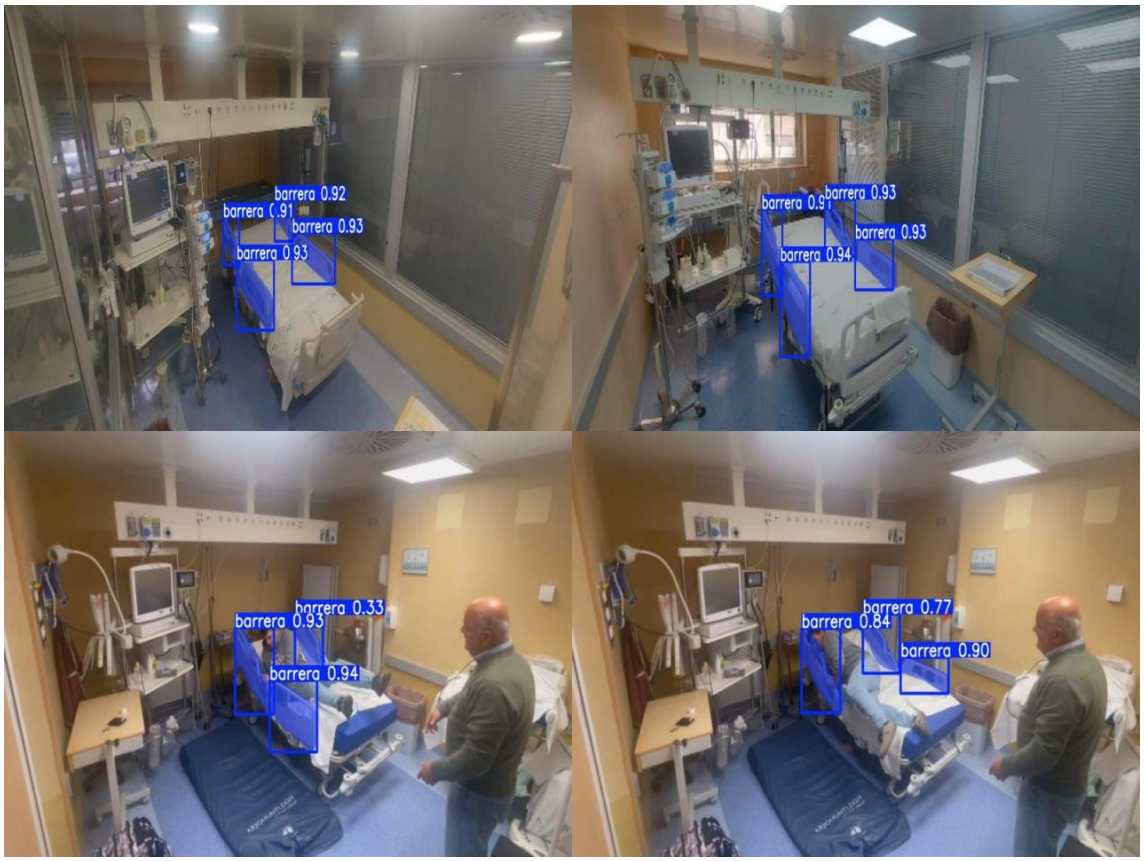


Figura 25. Pruebas prácticas de validación del segundo entrenamiento. Elaboración propia.

Se puede observar una notable mejora en los datos teóricos experimentales extraídos de estas gráficas. Se ve como los errores se disminuyen a niveles bastante más aceptables, aunque sigue habiendo deficiencias en la segmentación.

Las pruebas prácticas arrojan algo más inquietante, el modelo no es capaz de detectar las barreras con una persona delante. Hemos de atacar esta deficiencia.

1.3.2.3.4 Tercer entrenamiento.

En un tercer entrenamiento se termina de etiquetar el montante de 140 imágenes, haciendo hincapié de contar con la variabilidad que permita tener una persona delante de la barrera para tener cubierta esa contingencia. El entrenamiento tienes las mismas características que sus predecesores. Dada las características de mayor tamaño y en consecuencia un mayor requerimiento de GPU y tiempo de cómputo, se decidió limitar el número de épocas a 150, manteniendo el resto de características iguales a los entrenamientos anteriores.

Las gráficas obtenidas para este entrenamiento las podemos ver en la Figura 26 y Figura 27.

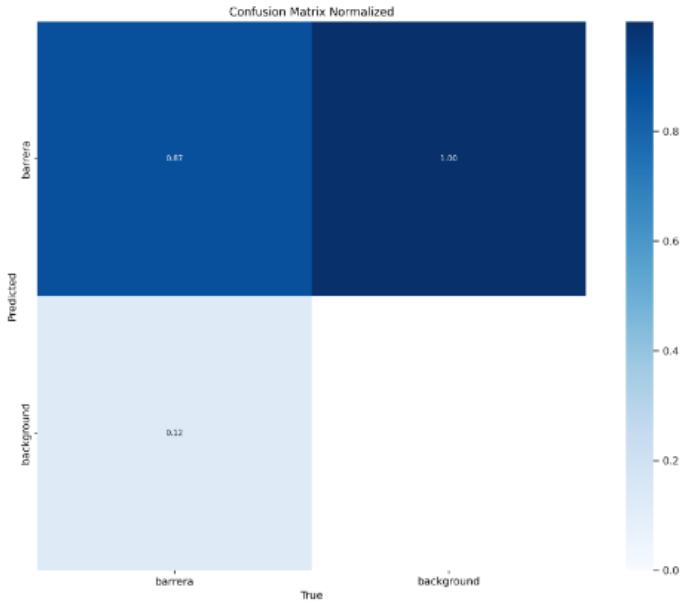


Figura 26. Matriz de confusión normalizada del tercer entrenamiento. Elaboración propia.

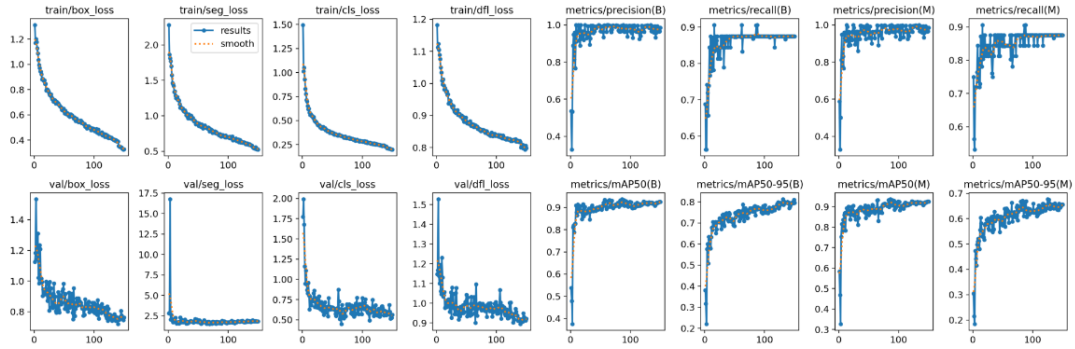


Figura 27. Métricas de validación del tercer entrenamiento. Elaboración propia.

En las gráficas de la figura, entrenado con menos épocas, se observa como el modelo va hacia un punto común, pero no alcanza la convergencia, y podría llegar a ser mejor y más preciso. Por ello, se decide realizar un cuarto entrenamiento para solventar esta característica.

1.3.2.3.5 Cuatro Entrenamiento.

En este entrenamiento se vuelve a configurar todos los valores de la misma forma que los iniciales, incluidas las 300 épocas. Así, se busca que el modelo se dirija hacia una zona de convergencia. Los resultados dados de esta cuarto y último entrenamiento son esclarecedores.

Las gráficas obtenidas para este entrenamiento las podemos ver en la Figura 28 y Figura 29.

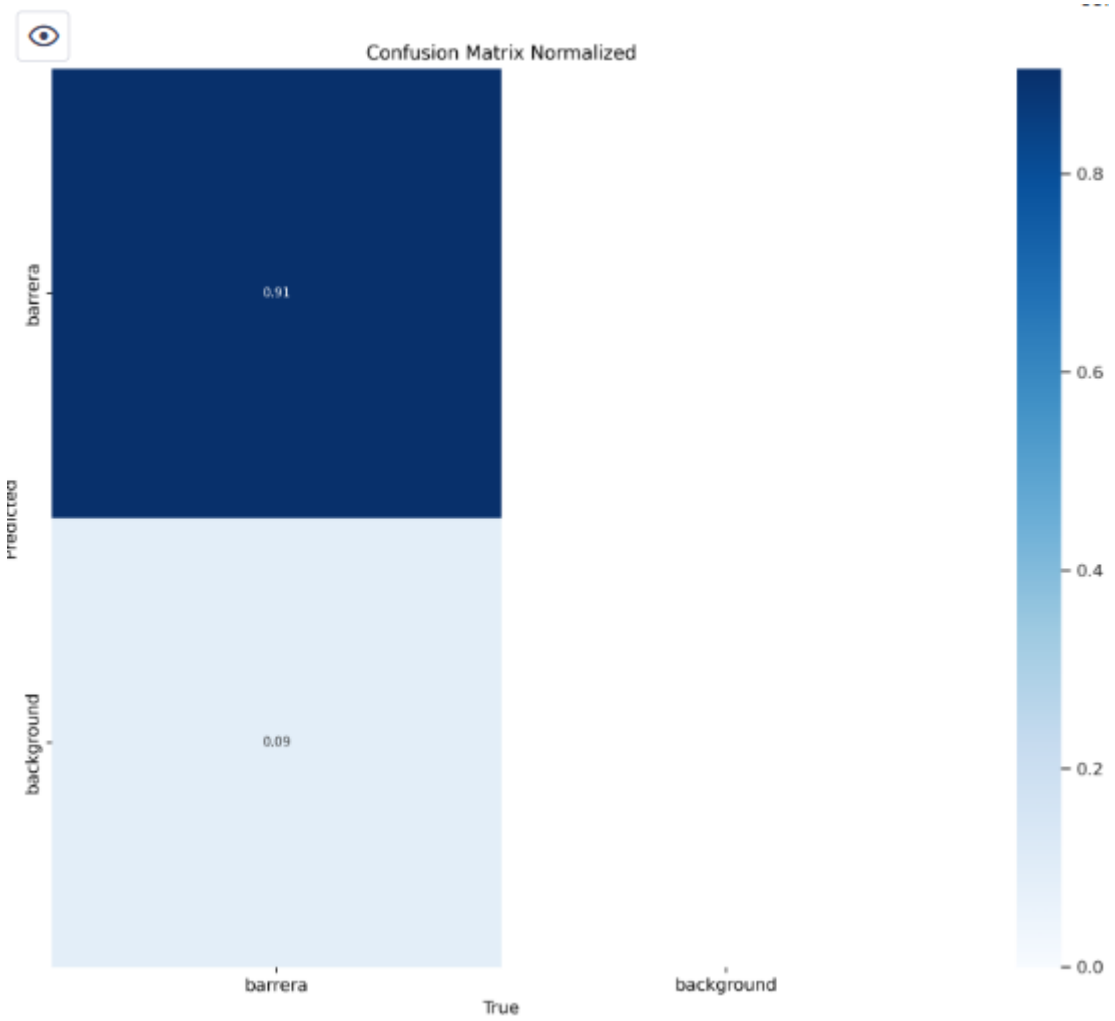


Figura 28. Matriz de confusión normalizada del cuarto entrenamiento. Elaboración propia.

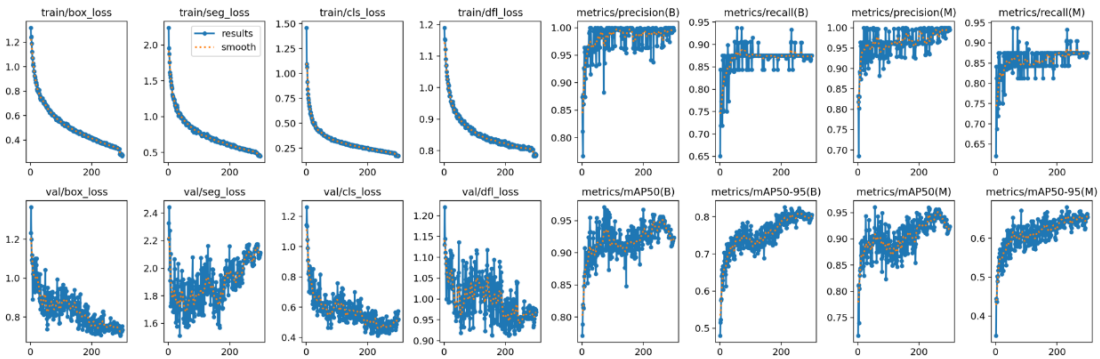


Figura 29. Métricas de validación del cuarto entrenamiento. Elaboración propia.

Las pruebas prácticas para este entrenamiento las podemos ver en la Figura 30.

Figura 30. Pruebas prácticas de validación de cuatro entrenamientos. Elaboración propia.

Se puede comprobar que este modelo mejora con mucho los anteriores y hace las detecciones de una forma lo suficientemente precisa como para ser estable y viable. Se asume que las fluctuaciones de las gráficas se deben a la augmentación y son inevitables. Se establece este modelo en producción.

1.3.3 Modelo de detección de individuos.

1.3.3.1 Objetivo del modelo.

Este modelo no fue inicialmente planteado al inicio del desarrollo. Cuando se comenzó el planteamiento y las pruebas de viabilidad de este, se vio la necesidad de poder discernir si una persona dentro del campo de visión de la cámara es un paciente o un trabajador sanitario. Así, dadas las características del proyecto, se llegó a la conclusión de que, si una persona se encontraba tumbada, dicha persona debía ser un paciente, y si se encontraba de pie, un sanitario.

Así, nos encontramos con una tipología de dataset un poco especial. Este tipo de datasets son muy generalistas. Se requiere un modelo que sea capaz de discernir bajo cualquier tipo de circunstancia si una persona está de pie o no, con toda la variabilidad que ello conlleva. Esa tarea requiere la recolección y etiquetado de un gran volumen de datos. Por

ello, me valí de un proyecto anterior realizado con mis compañeros de Siali, donde se recopiló una extensa cantidad de información y se trabajó en su etiquetado.

1.3.3.2 Entrenamiento y validación.

1.3.3.2.1 Modelo de detección.

Este proyecto nació en una doble vertiente. En un momento inicial se comenzó trabajando en el desarrollo de un modelo de detección. Para poder desarrollar correctamente un modelo de detección, es necesario etiquetar todas las personas que haya en cada imagen y determinar su posición.

Esa fue la tarea inicial y así se hizo. El entrenamiento fue de muy largo recorrido, ya que la cantidad de imágenes requerida era extraordinariamente grande, con más de 8000 muestras, por lo que se extendería por mucho tiempo. Este entrenamiento se realizó utilizando un modelo L, al requerir una capacidad de generalización extremadamente grande. Las características generales del entrenamiento se pueden ver en la Figura 31.

ARGS

aug_parameters	""
aug_parameters_states	""
augmentation	False
base_model	yolov8l.pt
batch_size	-1
clearML_download	True
clearML_project_name	Siali_proyecto_deteccion_caidas_fallen_sit_...
clearML_queue_name	train
clearML_task_name	caidas100L
dataset_path	/app/datasets/184/train.yaml
epochs	100
id_dataset	184
img_size	640
save_period	-1

Figura 31. Parámetros de entrenamiento del primer modelo de detección. Elaboración propia.

El modelo puede a priori llevar a confusión, ya que las gráficas de validación tienen una forma esperanzadora, como vemos en la Figura 32.

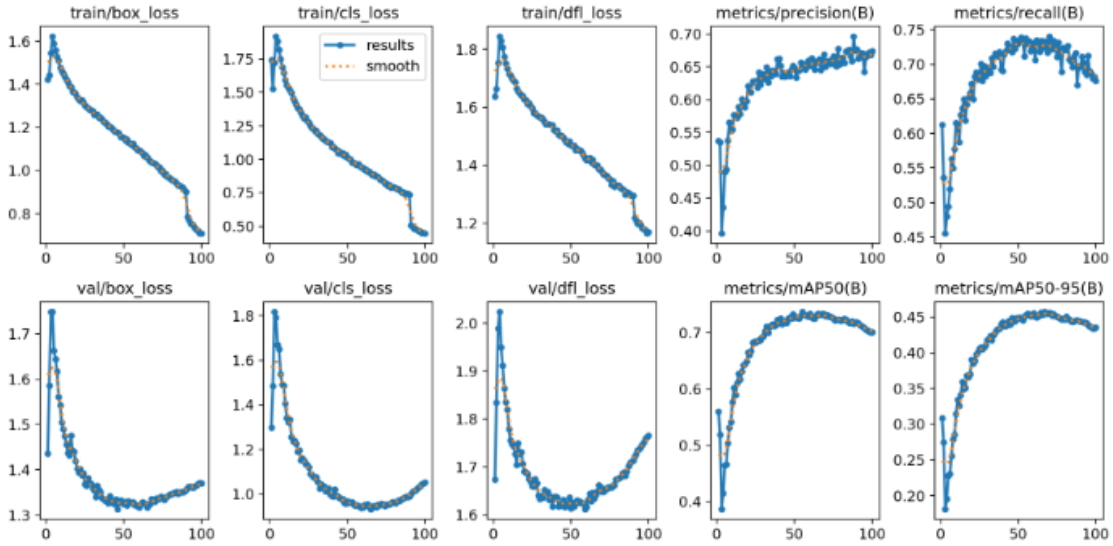


Figura 32. Métricas del primer modelo de detección. Elaboración propia.

Como se puede observar, ambas contienen una forma que indican que ha ocurrido un overfitting, especialmente en las gráficas más relevantes para este fin, que incluyen el box loss y el class loss, además de las mAP. El overfitting ocurre cuando un modelo ha llegado a su punto de máxima convergencia, por lo que todo paso que se dé en adelante, es decir, cada época que se realiza desde ese momento, como mínimo, añadirá un error de 0 a la mejor solución ya obtenida, y que, en la práctica, añade un pequeño error en cada iteración, ya que el punto de convergencia ya ha sido ubicado. Habitualmente, tener un ligero overfitting al final del entrenamiento es algo positivo, ya que asegura haber hallado la convergencia y no haber desperdiciado demasiado tiempo de cómputo en el proceso. En este caso, el overfitting llega en la época 50, por lo que significa que se ha perdido más de la mitad del tiempo de cómputo y eso es excesivo. Así, los modelos entrenados como se puede ver en la Figura 31 tienen un “save period” de -1, lo que indica que solo se guardarán los mejores pesos. En la práctica, esto significa que este entrenamiento ofrece los mejores pesos posibles para este dataset.

Sin embargo, si hacemos un análisis un poco más detallado de lo obtenido en este entrenamiento, se puede observar que tanto los errores de las pérdidas son muy elevados, por encima de 1, como que las métricas de los mAP son verdaderamente malas. Además,

las matrices de confusión hacen una lectura real de los errores de este, como se puede ver en la Figura 33 y Figura 34.

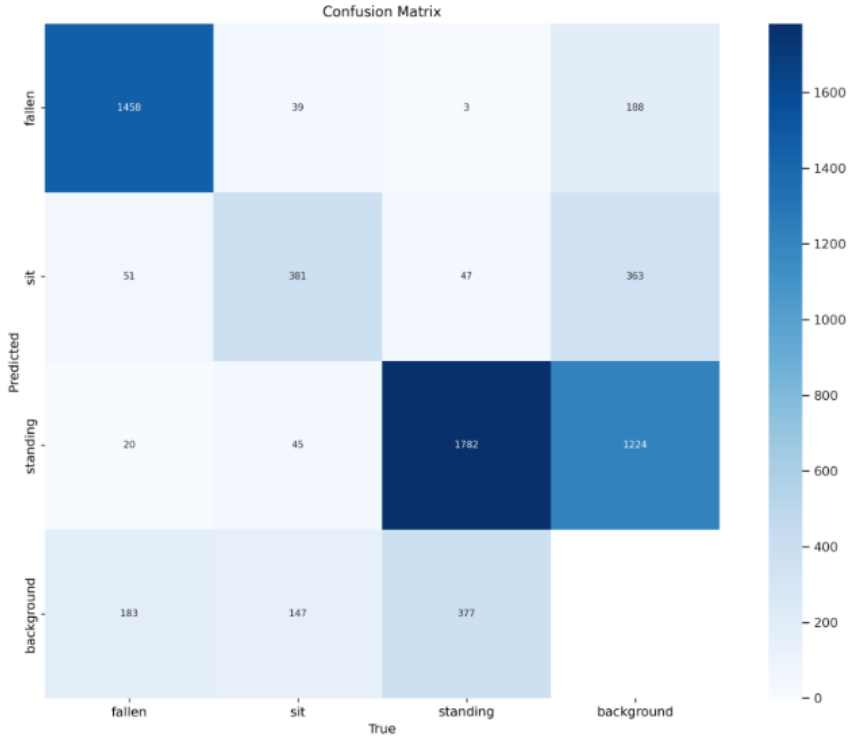


Figura 33. Matriz de confusión no normalizada del primer modelo de detección. Elaboración propia.

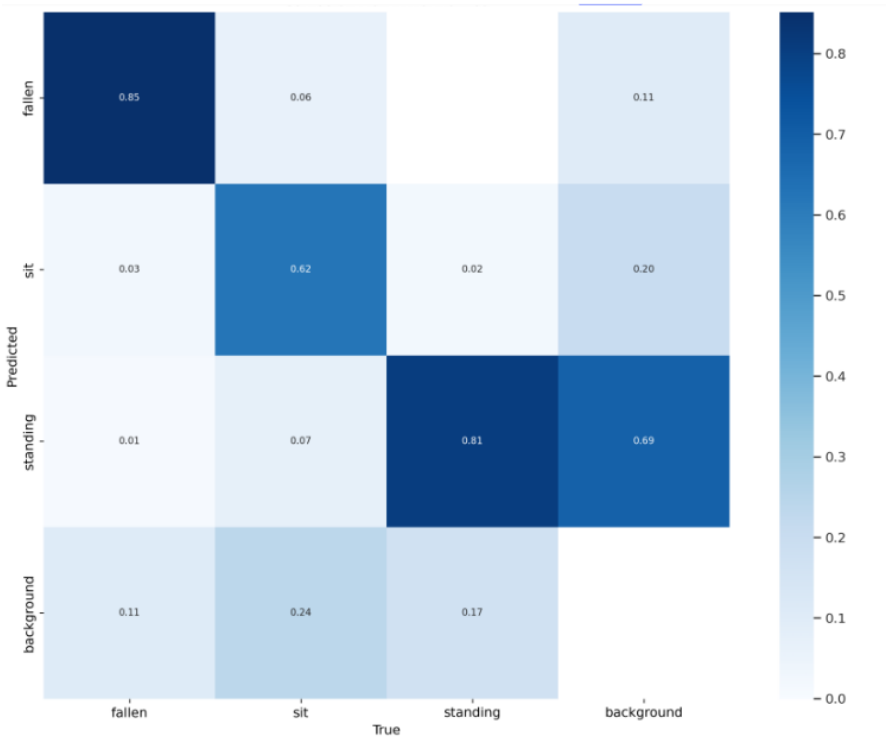


Figura 34. Matriz de confusión normalizada del primer modelo de detección. Elaboración propia.

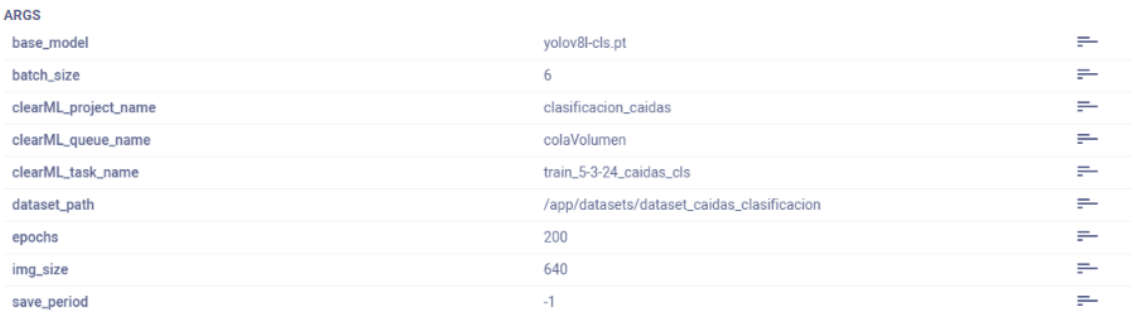
En conclusión, dado que el resultado obtenido era el mejor que podía ofrecer el dataset, esta línea de trabajo no parece la más indicada, por lo que se pasó a abordar en problema desde otra perspectiva.

1.3.3.2.2 Modelo de clasificación.

Tras barajar las posibilidades para la creación de un modelo generalista resistente a estas dinámicas, se oteó la posibilidad de crear un modelo de clasificación. Un modelo de clasificación de forma general analiza una imagen y devuelve un solo resultado para la imagen, con la confianza del modelo para que dicha imagen corresponda a cada clase. Es decir, deben definirse una serie de clases, y cada imagen solo debe adscribirse a una única clase.

En el plano práctico de entrenamiento, el proceso de etiquetado se realiza configurando tantas carpetas como clases presentes haya e introduciendo en ellas las imágenes. En el caso dado, se crearon tres carpetas, sentado, tumbado y de pie. Como el modelo pose ya otorga una detección de la persona, es posible pasarle dicho recorte a este segundo modelo, y será este el que realizará la verificación de la posición.

Así, se consiguió configurar las carpetas con un total de más de 16000 muestras. Un dataset bastante solvente y con una variabilidad elevada para alcanzar una buena generalización. Así, se eligió un modelo L y las características propias del entrenamiento se pueden comprobar en la Figura 35.



ARGS

base_model	yolov8l-cls.pt
batch_size	6
clearML_project_name	clasificacion_caidas
clearML_queue_name	colaVolumen
clearML_task_name	train_5-3-24_caidas_cls
dataset_path	/app/datasets/dataset_caidas_clasificacion
epochs	200
img_size	640
save_period	-1

Figura 35. Parámetros del modelo de clasificación. Elaboración propia.

Dado que es un dataset de clasificación, no aplican las gráficas clásicas ni las matrices de confusión vistas previamente, ya que al final, solo existen dos métricas que rastrear: que la clase con más confianza otorgada coincida con la real y la cuantificación del error (no es lo mismo fallar por un poco que fallar por mucho). Así, se pueden ver representadas ambas métricas en la Figura 36.

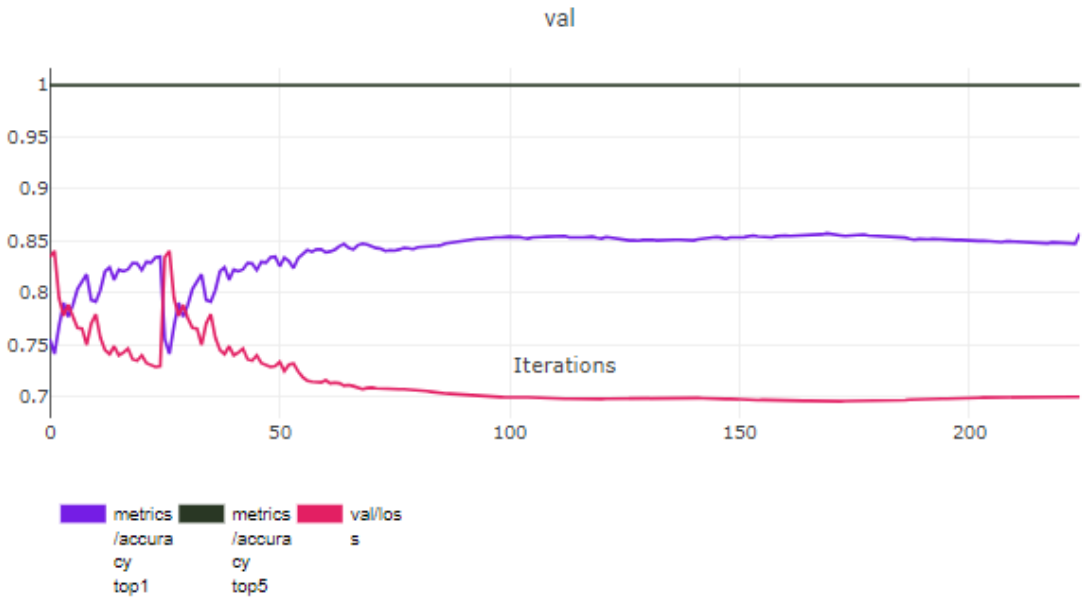


Figura 36. Métricas del modelo de clasificación. Elaboración propia.

Se puede observar cómo los resultados obtenidos son bastante buenos, con un error por debajo de 0,7. Se decide implementar este modelo.

2 Planificación y presupuesto

2.1 Planificación y hoja de ruta del proyecto.

La planificación del proyecto se pretende implementar metodologías ágiles, utilizadas en la conformación de proyectos relacionados directamente con el mundo digital y de desarrollo de software.

Estas metodologías se basan en la flexibilidad y la cooperación de forma continua con el cliente, así como la entrega de forma escalonada de servicios o partes del producto al cliente. Así, para esta tipología de planificación es ideal para la tipología de proyecto presentada.

La idoneidad de esta elección se basa en la naturaleza propia del proyecto. Los proyectos relacionados con inteligencia artificial dependen de muchas variables, pero sin lugar a duda el entrenamiento y la consecución de modelos completamente viables constituyen una de ellas. Como se vio antes, repetir y reiterar al obtener un modelo viable es parte fundamental de cualquier modelo de visión artificial. Es una dinámica habitual para alcanzar un buen modelo el reentrenamiento y la introducción de más imágenes al sistema.

Así, la validación del sistema y otras cuestiones relacionadas con el cumplimiento de los requerimientos del trabajo requieren una comunicación fluida entre cliente y equipo de desarrollo, ya que en proyectos ambiciosos con dificultades técnicas relevantes es importante aunar esfuerzos para conseguir objetivos comunes.

Debido a estas características es fundamental una dinámica de cooperación continua entre ambas partes, donde se recomienda reuniones periódicas de seguimiento en las que ambas partes tengan responsabilidades y objetivos a cumplir.

El proyecto se estructura de forma fundamental en cinco fases, con sus consiguientes subtareas:

1. Estudio previo.

2. Instalación de hardware y recopilación de datos.
3. Desarrollo de lógica y entrenamientos.
4. Deployment.
5. Validación.

Se pasará a detallar cada una de estas fases.

Fase I. Estudio previo.

En esta fase la misión fundamental es establecer los requisitos fundamentales para la viabilidad del proyecto. Así, se realizarán estudios sobre la necesidad de instalar cámaras y otros dispositivos auxiliares como iluminación, si es necesario. Además, otros dispositivos auxiliares como soportes también deben ser considerados en esta fase. Se debe definir la posición exacta de las cámaras, o en caso de que ya haya cámaras en el lugar de trabajo, si cumplen los requisitos necesarios para la viabilidad del proyecto.

Las unidades de cómputo también deben ser dimensionadas en esta fase. Es habitual en este tipo de proyectos que no se realizan en un entorno estrictamente industrial recurrir a sistemas on-cloud para el cómputo del software. Si este es el caso, se debe garantizar que la conectividad y la naturaleza de los servidores son adecuadas y ofrecen garantías para la viabilidad del proyecto.

En esta fase se deben definir las obligaciones del cliente sobre el mantenimiento general del proyecto, incluyendo en este apartado factores como mantenimiento periódico de cámaras y equipos, seguridad de estabilidad de la red, etc.

En este proyecto no aplica, pero se deberían garantizar las redes de comunicación, si es necesario, para iniciar actuadores, como alarmas sonoras externas.

Fase II. Instalación de hardware y recopilación de datos.

En esta fase, con ya toda la estructura física del lugar de trabajo definida, se debe instalar los equipos debidamente en la zona de trabajo. La instalación incluye el cableado y

conexionado de red, así como garantizar una conexión estable y segura, y la correcta emisión y recepción de las señales necesarias.

Así, con la puesta en marcha de equipos, se puede proceder a la verificación de la correcta recepción tanto de imágenes como de señales. Si la recepción de imágenes es correcta, se recopilan imágenes para entrenamiento.

Cabe destacar que la tarea de recepción de imágenes tiene dos peculiaridades. Por un lado, es una tarea que es difícil de definir en el tiempo, ya que está supeditada a agentes externos, donde destaca de forma clara la dinámica de trabajo de cliente. En este caso, depende, por ejemplo, de la ocupación de la habitación, por lo que la comunicación constante con el cliente es fundamental. Por otro lado, esta dinámica de recopilación de imágenes es muy probable que sea recursiva, por lo que la finalización de la etapa inicial de recopilación no implica el final del proceso.

Fase III. Desarrollo de lógicas y entrenamientos.

Así, llegamos a la fase de desarrollo. En esta fase el equipo de desarrollo comienza en paralelo diversas acciones. Por un lado, comienzan los entrenamientos de los modelos, donde es habitual recurrir a departamentos expertos en la materia para su calibración.

Se realiza en paralelo el desarrollo del código así como la construcción completa del sistema. La comprobación y validación del desarrollo constituye parte constante del proceso.

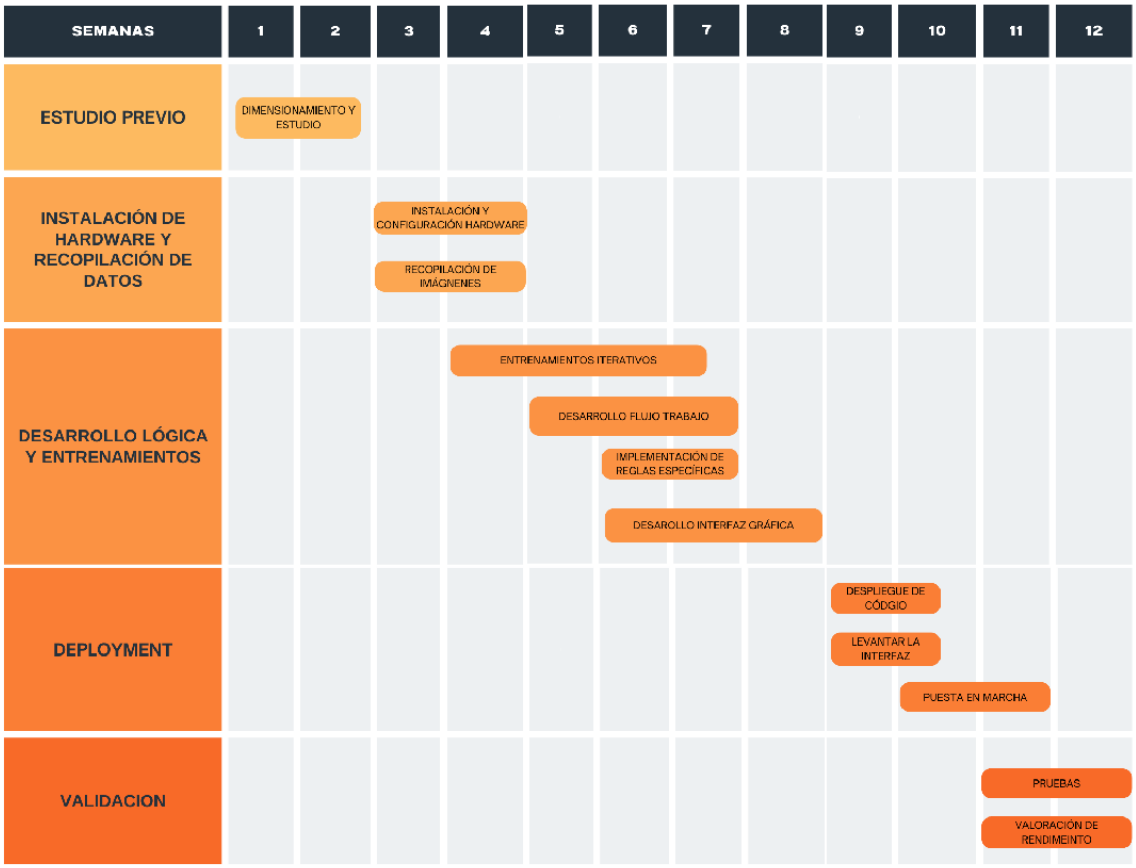
Fase IV. Deployment.

Una vez finalizado el desarrollo del programa, se procede al despliegue de este en el lugar de trabajo. Constituye una fase vital donde los ajustes y las correcciones a raíz de la integración del software con los sistemas presentes son especialmente relevantes.

Fase V. Validación.

Con el sistema instalado y funcionando, se realizan protocolos de seguimiento y comprobación del software, verificando que su funcionamiento es correcto. Durante esta fase, el sistema todavía no cuenta como herramienta de respaldo para el cliente, por lo que podríamos decir que no está en producción. Una vez validado el producto, es entregado al cliente con todas las garantías.

2.2 Diagrama de Gantt



2.3 Presupuestos

Predicción de caidas

Hospital

Referencia: 20240909-150519459
Creación del presupuesto: 9 de septiembre de 2024
Caducidad del presupuesto: 9 de septiembre de 2025
Presupuesto creado por: Ismael Barakat Fernández

i.barakat@icuwatch.com

Productos y servicios

Artículo y descripción	Cantidad	Precio unitario	Total
POC			
Hardware POC - 1 Cámara - 1 Computo - Cables y accesorios	1	4.200,00 €	4.200,00 €
Instalación de 1 cámara	1	1.100,00 €	1.100,00 €
Despliegue e integración de Icuwatch Despliegue e integración del sistema en las instalaciones del cliente	1	3.300,00 €	3.300,00 €
Proyecto completo			
Licencia plataforma Icuwatch (por planta) **El pago comienza: 6 meses después del pago inicial**	1	8.000,00 € /año	8.000,00 € /año

Artículo y descripción	Cantidad	Precio unitario	Total
Instalación 6 cámaras La instalación la realizara un proveedor externo **Vencimiento del pago: 6 meses después del pago inicial**	6	1.100,00 €	6.600,00 €
Licencia por cámara (después del escalado) Licencia por cámara instalada de forma permanente en las instalaciones del cliente **El pago comienza: 6 meses después del pago inicial**	7	250,00 € /año	1.750,00 € /año
Desarrollo e integración	1	18.500€	18.500€
	POC		8.600,00 €
		Total	**34.850€**

Condiciones de compra

La licencia de Icuwatch, cubre asistencias y mantenimientos en remoto, 30h de formación para autonomía del empleado, corrección de bugs y actualizaciones de mejoras.

Una vez puesto en producción todo aquel problema que no venga derivado del software, se presupuestara a parte por bolsa de horas.

Las condiciones para dar por cerrado el desarrollo e integración de Icuwatch que dan paso al escalado, de la siguiente cotización, serán acordados con el cliente y definidos con el nombre "Condiciones cierre Hospital" mediante un documento o correo electrónico.

El frontend/backend a gusto del cliente, con una personalización por parte de producto, no contemplada en el roadmap por parte de Icuwatch , será cotizado como horas de consultoría a parte.

Facturación del total del proyecto:

Factura 1 (al recibir el pedido): 100%Hardware+articulos que vencen ya (POC)

Factura 2 (Artículos que vencen mas adelante): 100% Proyecto

3 Especificación

3.1 Estudios y Análisis Previos.

Se realiza estudio sobre la viabilidad y requerimientos del proyecto. El objetivo es conseguir la viabilidad del sistema y predecir las caídas en una habitación como primario.

Por parte del equipo de desarrollo, se define que se requiere unos medios que permitan gestionar una cámara en tiempo real, así como tres modelos de visión artificial. Los tres modelos corresponden a la clasificación del sujeto en paciente u otro, la segmentación de las barreras de la cama y el posicionamiento de puntos clave del sujeto.

En proyectos de esta índole es habitual el requerimiento por parte del cliente del borrado de caras de los agentes involucrados las imágenes. Esta dinámica sumaria un modelo de visión anexo. Dado que las imágenes obtenidas no se van a guardar, ya que son desechadas instantáneamente y solo pueden ser visualizadas por técnicos in situ, se desecha esta vía.

La cámara se colocará en un plano alto, en una esquina de la habitación, con un ángulo suficiente para ver la cama del paciente, respetando cierto margen de seguridad para asegurar las detecciones.

De forma general, a falta de especificación más precisa en el Documento de Especificación, se requiere:

- Cámara CCTV.
- Unidad de cómputo. Debe contar con:
    - Tarjeta gráfica de, al menos, 12 GB.
    - Sistema operativo Ubuntu.
    - Drivers de Nvidia.

Se recomienda, dada la criticidad del proceso, que el equipo de cómputo constituya una unidad física cerca de la zona de trabajo y que no se constituya en un terminal en cloud.

3.2 Requisitos de usuario.

El usuario solicita un sistema capaz de predecir mediante patrones de movimientos de pacientes con ritmo de actuación lento la posibilidad de una caída.

Para ello, requiere instalar una interfaz gráfica que monitoree en tiempo real la situación del paciente, que se visualice de forma constante en la interfaz, y que haya una señal visual muy notable si la caída se produce.

El usuario requiere que el sistema tenga una tasa de detección mayor al 95% y no requiere el borrado de cara, ya que se garantiza que no existe almacenamiento de imágenes. Asimismo, se solicita que la tasa de refresco del proceso sea lo suficientemente elevada como para mantener la funcionalidad del sistema.

3.3 Documento de Especificación (IEEE 830-1998)

3.3.1 Introducción

El **objetivo** del proyecto es la implementación de un sistema capaz de predecir mediante patrones de movimientos de pacientes con ritmo de actuación lento la posibilidad de una caída.

El **alcance** del sistema está limitado a una herramienta de apoyo al equipo técnico, y no tendrá aparejada ninguna funcionalidad autónoma per se.

3.3.2 Descripción General

El producto monitorizará una habitación específica mediante la utilización de un cámara asistida por una unidad de cómputo. Dicha unidad utilizará tres modelos de visión artificial los cuales podrán asociar patrones de movimientos característicos que preceden a una caída. En caso de detectarse esta situación, una señal visual alertará al técnico responsable.

En relación con la **perspectiva** del producto, pese a estar configurado de forma inicial para la monitorización de una sola cámara en una sola habitación, está diseñado para, mediante una mejora de las especificaciones de la unidad de cómputo y ligeros cambios de arquitectura de software, poder operar un número indefinido de cámaras.

En relación con las **funciones** del producto, se comprende como un sistema de apoyo al técnico, cuya finalidad es permitir al responsable tener un apoyo para la monitorización y asistencia durante cargas de trabajo grande o en periodos de personal reducido.

En relación con las **características de los usuarios,** está orientado únicamente a técnicos sanitarios encargados de la supervisión de pacientes, en particular, enfermeros y auxiliares.

En relación con las **restricciones** del producto, estará constituido en Docker y programado en Python. No tendrá protocolos de comunicación con otros elementos externos al terminal y no requerirá de comunicación a la red para su funcionamiento.

3.3.3 Requisitos del sistema

Para su funcionamiento, se requiere que el sistema cuente con:

- Unidad de cómputo. Debe contar con:
    - CPU con capacidad de gestión elevada, 12th Gen Intel(R) Core (TM) i7-12650H o superior.
    - Tarjeta gráfica Nvidia 3060 o superior.
    - RAM 16 GB DDR 4 o superior.
    - SSD 500 GB o superior.
- Cámara CCTV con resolución HD.
- Python 3.10
- Docker
- Librerías anexas para la lógica (podrán ser verificadas por el equipo de IT).

Asimismo, requiere por parte del cliente:

- Protocolo de mantenimiento y limpieza de cámaras.
- Protocolo de mantenimiento de conexiones.
- Protocolo de mantenimiento del servidor.

3.3.4 Criterios de aceptación

Para que el producto pueda considerarse como viable y pueda en producción, con la consiguiente liquidación aparejada al último hito de validación, el sistema debe:

- Predecir de forma correcta al menos el 95% de las caídas. Esta tasa debe ser independiente de la velocidad del paciente y está expresada como objetivo de liquidación.
- Tras la liquidación se establece un ánimo de mejora continua con un objetivo cercano al 100% para la ampliación a más cámaras.
- El sistema debe tener una señal visual suficiente.
- La tasa de refresco del sistema debe permitir al técnico la monitorización el tiempo real del paciente.

4 Diseño e Implementación

4.1 Diseño (Preliminar y Detallado)

El diseño del sistema se estructura en forma de microservicios los cuales se hacen peticiones entre ellos. En este sentido, tenemos tres servicios diferenciados: la API de inferencia, el main y el front-end. Como no nos es necesario y además se incurriría en problemas de protección de datos, a la vez que complicaríamos de sobremanera el proyecto, hemos decidido prescindir de una base de datos o de otra forma de back-end. La gestión de recursos será directa y no será necesaria la implementación de microservicios anexos para su almacenamiento.

Así, tenemos una API que gestiona las inferencias y la relación con el YOLO. Mediante una línea de trabajo diferente a la habitual, se ha decidido que sea la API en su conjunto la que maneje la estructura general del proceso, desde las inferencias a la gestión de la cola de caídas, pasando por la lógica de predicción. Como ahondaremos más adelante, este enfoque permite un proceso constante e ininterrumpido. Este enfoque se aleja del uso habitual de mero intercomunicador que habitualmente se les da a una API, y gracias al framework que utilizamos, FastAPI, se le puede dotar de una utilidad mayor.

Así, la API se encarga del proceso general: inferencias, lógica y gestión de alertas. Streamlit es el encargado de ofrecer una interfaz gráfica al usuario a modo de front-end. El main se encarga de la gestión general de los servicios, así como hacer llamadas a los endpoint para que se ejecuten de manera escalonada.

Toda esta estructura de microservicios está gestionada por Docker, que como se verá más adelante en profundidad, ofrece un entorno estable para el despliegue, independiente del equipo donde se levante.

4.1.1 Algoritmos y lógicas de prevención de caídas

El algoritmo sigue un camino claro dentro de la API. La API está diseñada para realizar las inferencias en un orden concreto que se detallará más adelante, pero ese orden de inferencia no determina el orden que sigue la lógica. La lógica de la predicción, la parte

central sobre la que gira todo el proyecto, está definida por la función *check_fall*, y nos valdremos de la misma para explicar tanto la estructura general del código, como cómo gestionamos los resultados de la inferencia con modelos.

Es importante destacar que, como se ha explicado, las inferencias se realizan de manera anterior, por lo que ya se tienen los resultados como argumentos de entrada de la función. La única inferencia que se realiza dentro de esta dinámica es la de clasificación.

Se usará una misma secuencia recurrente para explicar los pasos dados, la cual está formada por cuatro frames escogidos justo en el momento en el que se pasa a predecir la caída. Así se podrá tener una visión completa del procedimiento. Se puede observar la secuencia en la Figura 37.

Figura 37. Secuencia a estudio. Elaboración propia.

4.1.1.1 Uso de Keypoints de persona.

El uso de Keypoints constituye el eje fundamental del proyecto y es donde se basa la parte técnica del mismo. Esta herramienta nos permite extraer todas las lógicas comentadas con anterioridad y relacionarlas con la posición relativa de la cama.

El uso del modelo *pose.pt* nos devuelve, además de los 17 puntos clave del individuo, una bounding box que determina la posición de la persona. De esta forma, nos es posible iterar entre las personas presentes en la sala para saber cómo analizar su estado. Así, se puede ver lo que devuelve este modelo en la Figura 38.

Figura 38. Secuencia con la inferencia del modelo pose. Elaboración propia.

4.1.1.2 Lógicas relativas a la posición del individuo.

El primer problema al analizar el estado del paciente es discernir si la persona a analizar es realmente un paciente, y no un sanitario u otra persona en la sala, como una visita. Así, mediante el modelo *pose.pt*, se localiza la cantidad de personas en sala. Haciendo uso de las bounding boxes, que no de los Keypoints, se itera sobre la detección, recorriendo las bounding boxes de persona detectadas.

Se realiza un recorte de la imagen según la detección de persona obtenida, y esta es inferenciada mediante el modelo posicion.pt. Este modelo devuelve una respuesta única, con la cual enumera la confianza de que el individuo esté de pie, sentado o caído. Se puede observar estos resultados en la Figura 39.

Figura 39. Secuencia con la inferencia del modelo de posición. Elaboración propia.

Así, se presupone que el individuo tumbado será el paciente, y si está de pie o sentado, no. Mediante un condicional, el algoritmo solo continúa con las detecciones de aquellas personas tumbadas.

4.1.1.3 Lógicas relativas a la detección de la cama.

Una vez identificado el paciente, se pasa al análisis su estado a través de la función check_fall. Antes de entrar a la función, se recopilan todos los datos necesarios. En este caso, un diccionario formado por los Keypoints, de una manera fácil de utilizar, y otro diccionario formado por las áreas de las barreras detectadas con anterioridad mediante el modelo *barrera.pt.* Este modelo devuelve, además de la detección, la delimitación exacta del objeto. Se puede observar la detección para la secuencia de estudio en la Figura 40.

Figura 40. Secuencia con inferencia de modelo de barreras. Elaboración propia.

Al tener toda esta información, se pasa a la función de predicción. Básicamente, se ha implementado una lógica mediante la cual, si al menos tres de los Keypoints correspondientes a ‘muñeca’, ‘rodilla’ y ‘tobillo’, se encuentran dentro de las barreras al mismo tiempo, se genera una señal de caída. Como se verá después, una sola señal de caída no es suficiente para activar la alarma, para evitar falsos positivos.

En la Figura 41, se puede observar cómo antes de la caída, en posición de reposo normal, los Keypoints se encuentra en una posición correcta, y cuando el paciente intenta aproximarse y realizar una salida, los Keypoints entran en zona y la señal caída se activa. Esta imagen es muy indicativa porque se ven representadas todas las detecciones relevantes para la señal.

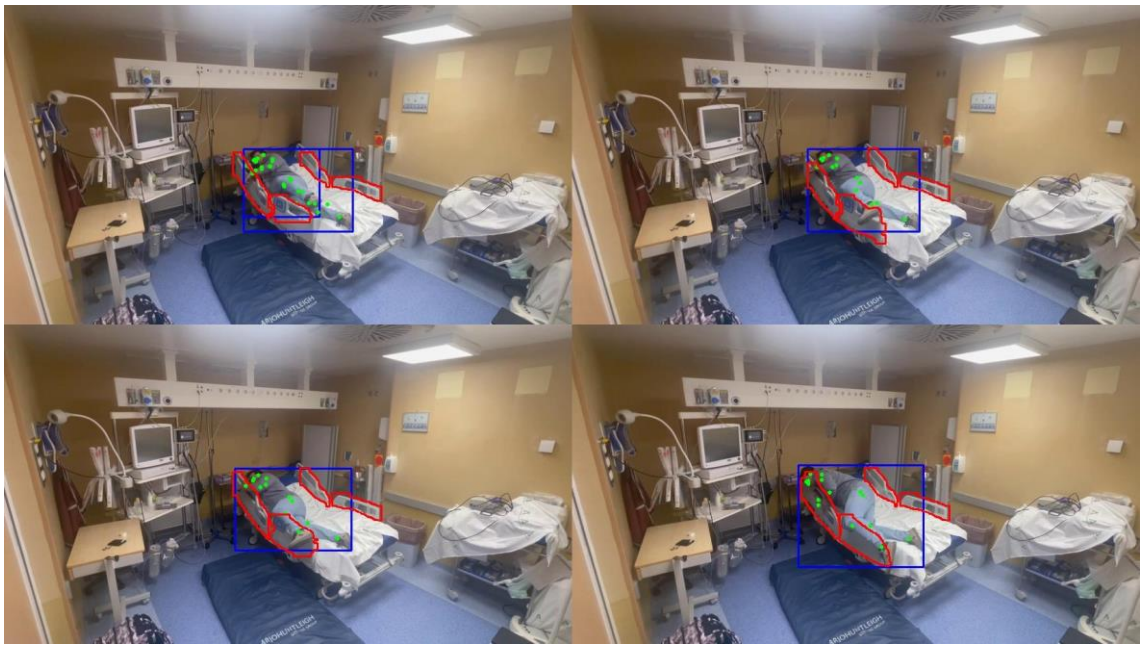


Figura 41, Secuencia con representación gráfica de modelos pose y barreras. Elaboración propia.

Asimismo, esta dinámica aporta beneficios, ya que la posición del individuo para activar la señal es indicativa y no es fácil que se dé de forma natural. De la misma forma, si, como se ha comentado con anterioridad, estos pacientes se encuentran semiinconscientemente, habría tiempo para asistirles antes de una caída.

4.1.2 API de inferencia.

4.1.2.1 Objetivo y por qué de la API.

Una API es una herramienta que permite la comunicación entre servicios sin necesidad de tener que conocer detalles internos del otro para la comunicación. En la industria se usa por diversos motivos, como su estabilidad, su fácil integración y su modularidad.

En este caso, la utilización de la API se basa en dos cualidades muy interesantes de esta herramienta. Por un lado, permite la interoperabilidad de sistemas que están estructurados en frameworks y lenguajes diferentes. En este sentido, las llamadas a la API se realizan mediante el denominado cURL, el cual puede ser traducido a cualquier lenguaje de programación, por lo que independientemente del lenguaje utilizado se pueden realizar llamadas a la API.

Este factor es muy relevante con relación a la segunda cualidad interesante para este proyecto: la escalabilidad. Dada esta característica, y la propia concepción de la API, el aumento del número de cámaras involucradas o el cambio del front-end, podría hacer que la modularidad del proyecto se mantuviera y que los cambios fueran mínimos.

Si, por ejemplo, se decidiera dar un salto de calidad y realizar un front-end más completo en React, por ejemplo, la API podría ser utilizada de igual forma.

En este caso concreto, se utilizó el framework de FastAPI, el cual permite construir la API de una forma muy sencilla en Python, sin perder rendimiento. Asimismo, soporta procesos asincrónicos, muy relevantes cuando se quieren gestionar recursos paralelos, como es este caso. El enfoque minimalista de este framework permite un desarrollo rápido y eficiente del mismo. Se puede observar la interfaz de la API en la Figura 42.

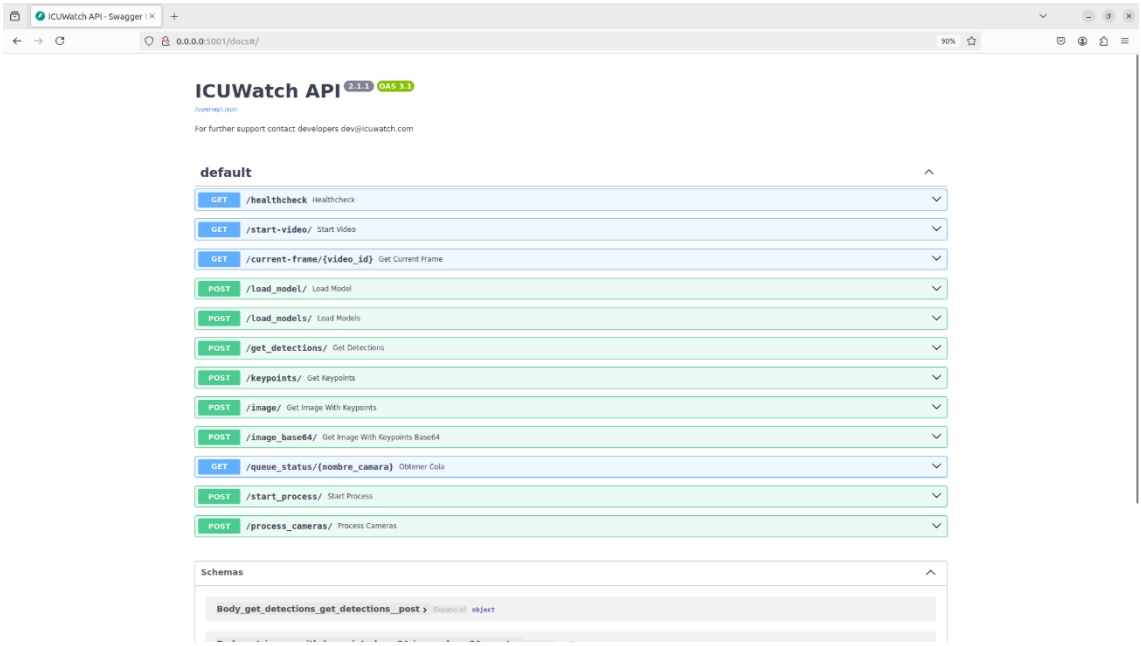


Figura 42. Swagger de la API. Elaboración propia.

4.1.2.2 Endpoints relevantes.

Se pasará a detallar el funcionamiento de los endpoints más relevantes de la API y que conforman la dirección del algoritmo. Como se ha comentado antes, esta API es la que realiza todo el proceso. El main solo gestiona la puesta en marcha del sistema, y el front-end hace peticiones recursivas a la API para obtener información y postearla.

Así, se hará un recorrido por los principales endpoints en orden de actuación detallando cómo funciona el sistema.

4.1.2.3 Start-video

Tiene como argumento de entrada un JSON que le especifica o bien la URL de entrada de la cámara, o bien una ruta a un video para trabajar en simulado. En este sentido, tanto en un caso como en otro, la función de este es cerciorarse de que el streaming de video, o el video en sí, se encuentran corriendo y generando frames que posteriormente puedan ser traspasados a la inferencia o al front-end. Se puede ver su presencia en la Figura 43. En el JSON se estructura de tal forma que se pueden incluir tantas cámaras como se desee, ya que la función está preparada para trabajar con varias cámaras de manera simultánea.

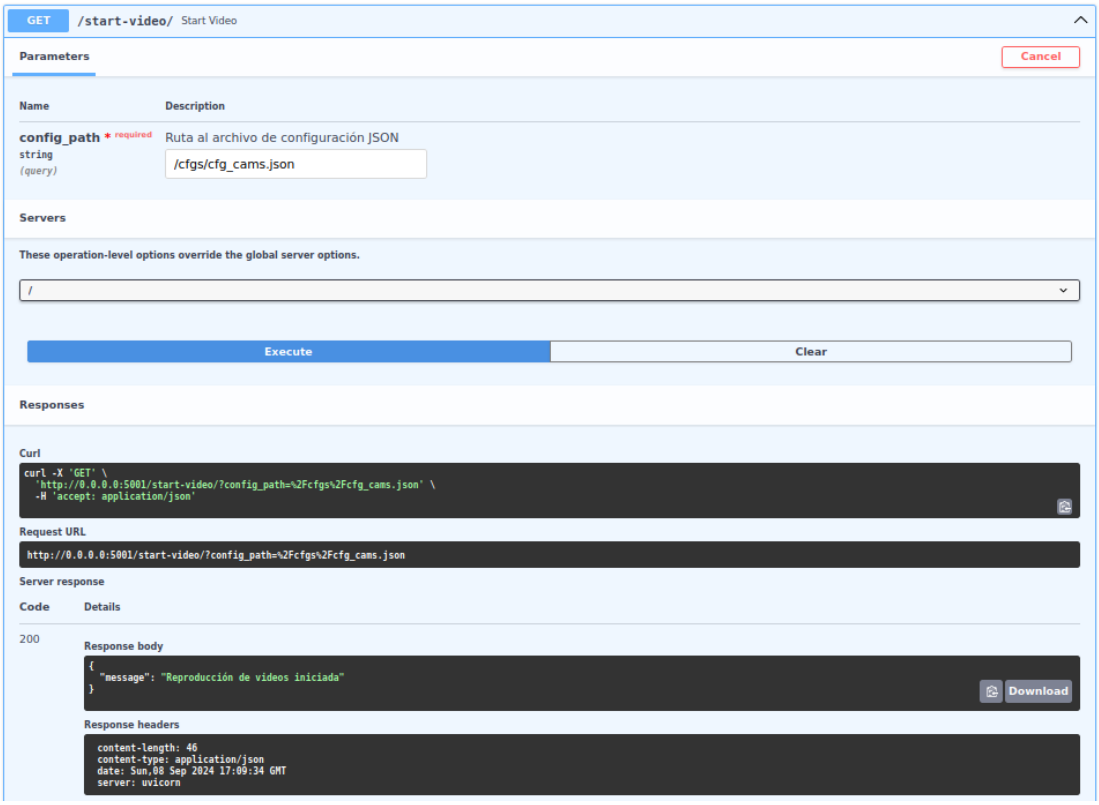


Figura 43. Endpoint de start-video en el Swagger. Elaboración propia.

4.1.2.4 *Current-frame*

Este endpoint es de vital importancia, ya que es el que proporciona las imágenes tanto que se retransmitan en el front-end, como que se inferencian para la predicción. En este sentido, su único cometido es proporcionar frames de manera continua, y devuelve las imágenes tras darle el nombre de la cámara, ya que está diseñada para trabajar con varias cámaras en paralelo. En la Figura 44 vemos interfaz gráfica.

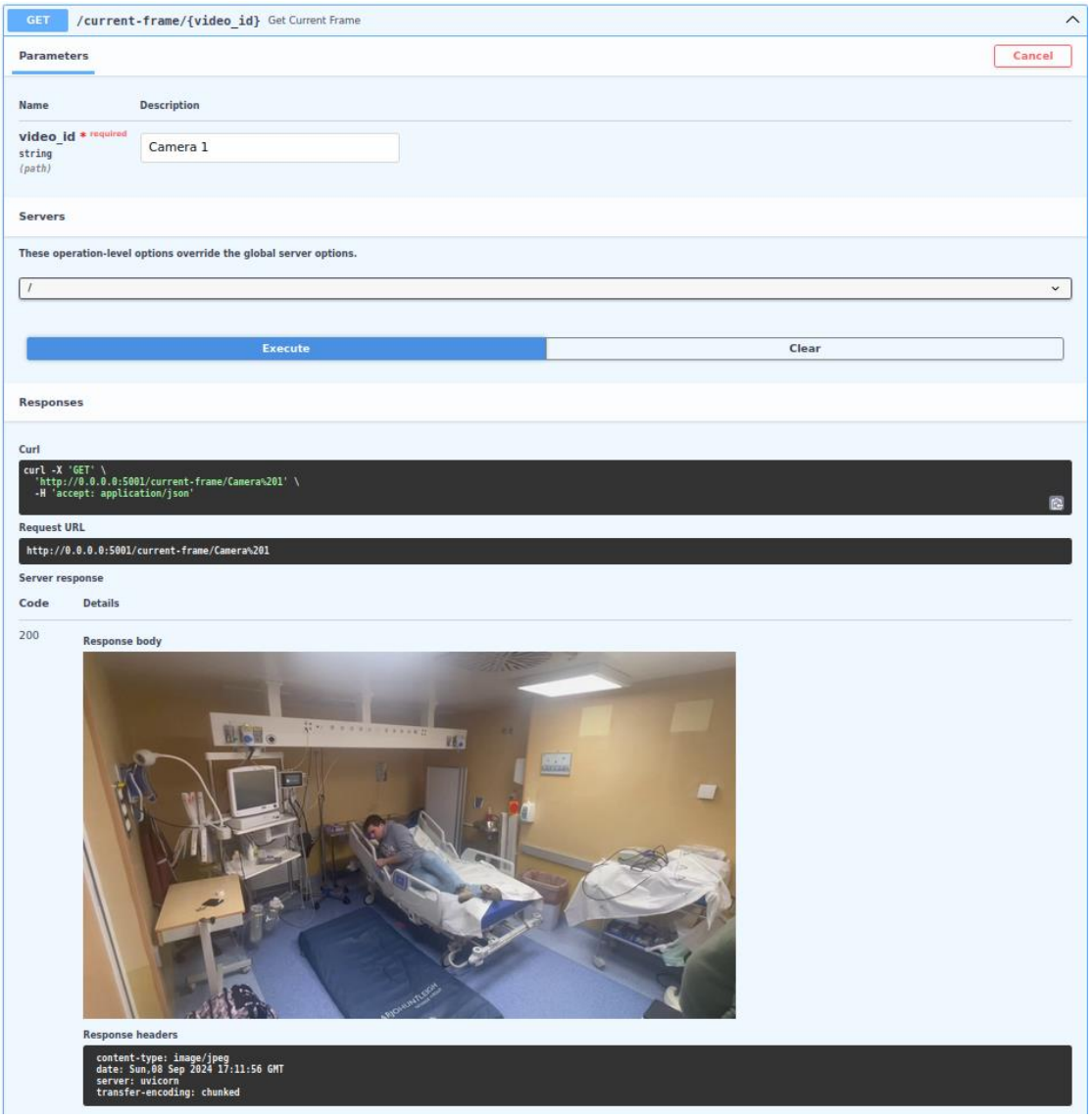


Figura 44. Endpoint de current-frame en el Swagger. Elaboración propia.

4.1.2.5 Load_models

Endpoint fundamental para inferenciar de una manera práctica y rápida. Su función es cargar los modelos de YOLO y dejarlos listos para inferenciar. Asimismo, este endpoint tiene un enfoque novedoso, ya que no carga los modelos de manera individual. Tradicionalmente, para inferenciar imágenes con YOLO se debía cargar cada modelo de manera individual y posteriormente inferenciar con él. Este enfoque requiere mucha memoria GPU, que el terminal que soporta el proyecto no dispone.

Un enfoque alternativo a esta dinámica consiste en la carga de un modelo para una inferencia, y posteriormente cargar otro para la siguiente inferencia. Trasladado al caso práctico, en este caso no es viable, ya que se necesita que la velocidad de procesamiento sea del orden de los milisegundos, y el tiempo de carga del modelo es exponencialmente mayor al tiempo de inferencia.

Por ello, se buscó un enfoque alternativo en el que los modelos se cargan en una lista, y quedan guardados en una variable global, que se puede usar en toda la API. Este enfoque consume menos de 1 GB de gráfica, como veremos más adelante, mientras que enfoques de carga individualizada podrían requerir hasta esa cantidad por modelo.

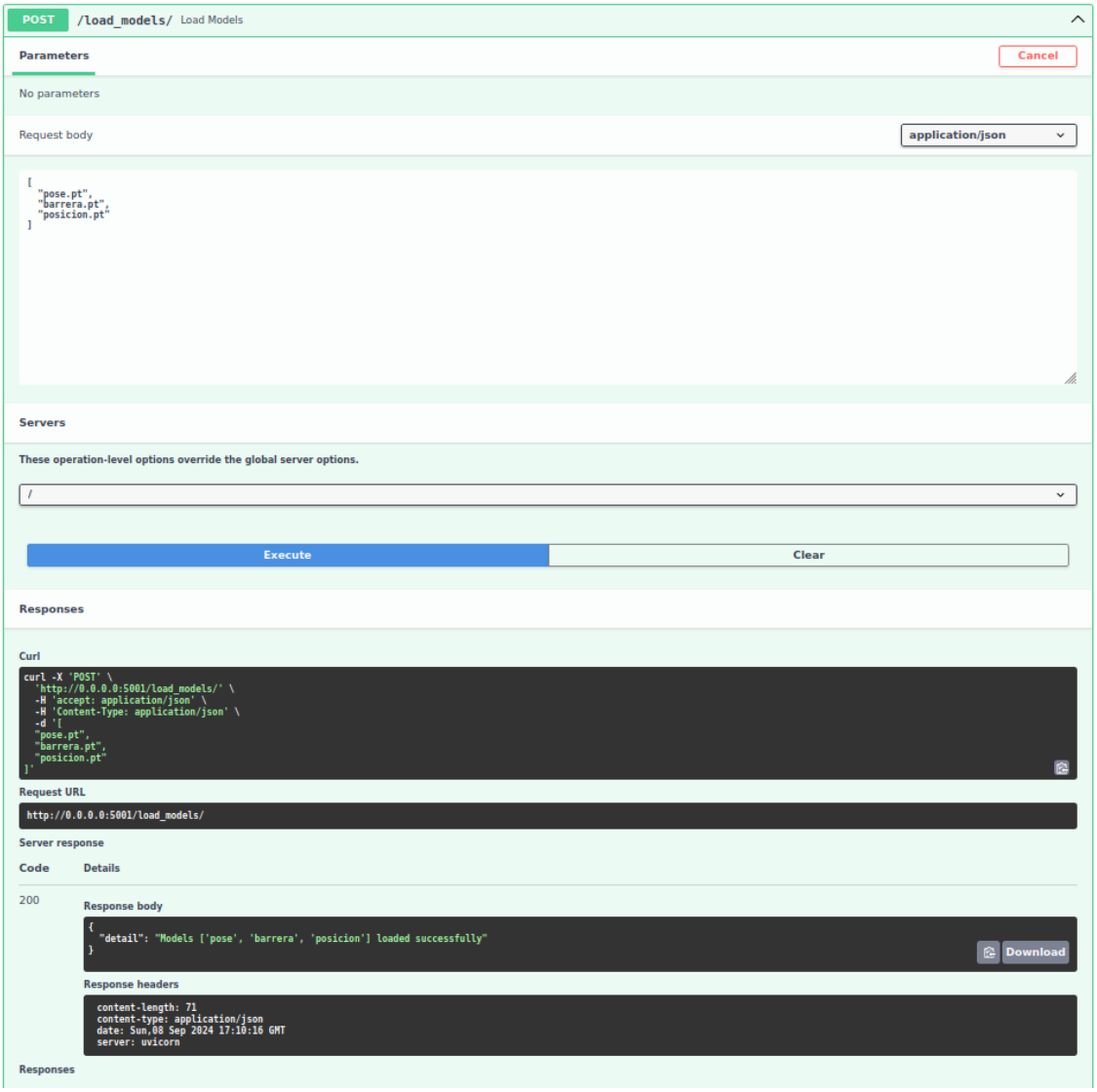


Figura 45. Endpoint de load_models en el swagger. Elaboración propia.

4.1.2.6 Process_cameras

Otro endpoint de vital relevancia, ya que su función es hacer peticiones continuas de frames e inferenciarlos con los modelos pose.pt y barrera.pt. También se encuentra preparado para usarse de manera extensiva con la implementación de varias cámaras. Esta dinámica se ve reflejada en la utilización de tres variables globales, las cuales en forma de diccionario guardan las detecciones de pose, las detecciones de la barrera y el frame en sí de cada cámara, siendo la clave de cada diccionario el nombre de la cámara. De esta forma se puede acceder de manera continua a los datos de cada cámara en cualquier momento.

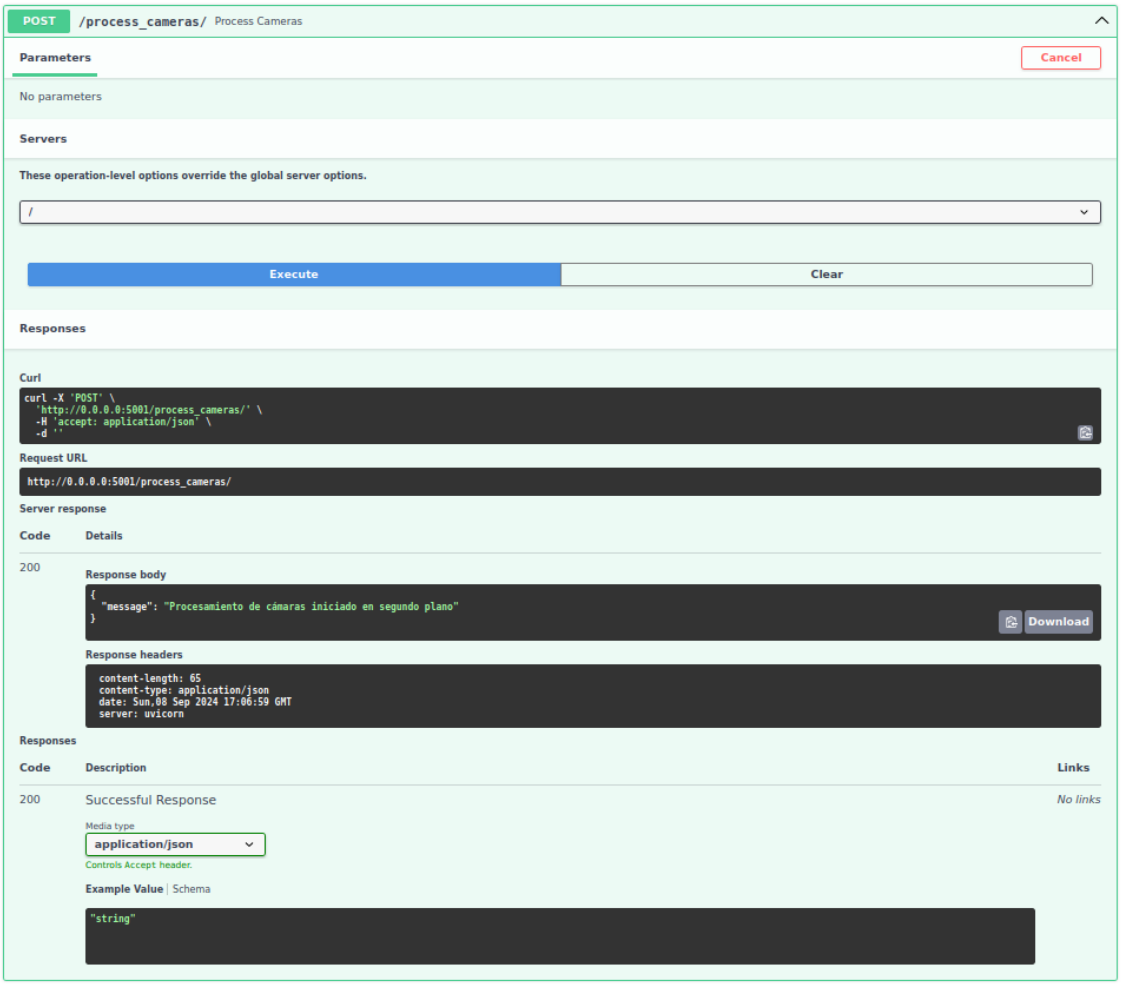


Figura 46. Endpoint de process_cameras en el swagger. Elaboración propia.

4.1.2.7 Start_process

Este endpoint abre nuevamente el JSON y de manera automática abre tantos hilos como cámaras haya disponibles. Los hilos se encargan de abrir la función *check_fall*, admitiendo como argumento de entrada el nombre de la cámara. La función recopila las variables globales en forma de diccionario y les introduce la clave de la cámara, la cual comienza a trabajar de manera recursiva y a dar resultados de la señal de caída. Posteriormente, comienza a guardar señales en la cola de esa cámara, como se explicará a continuación.

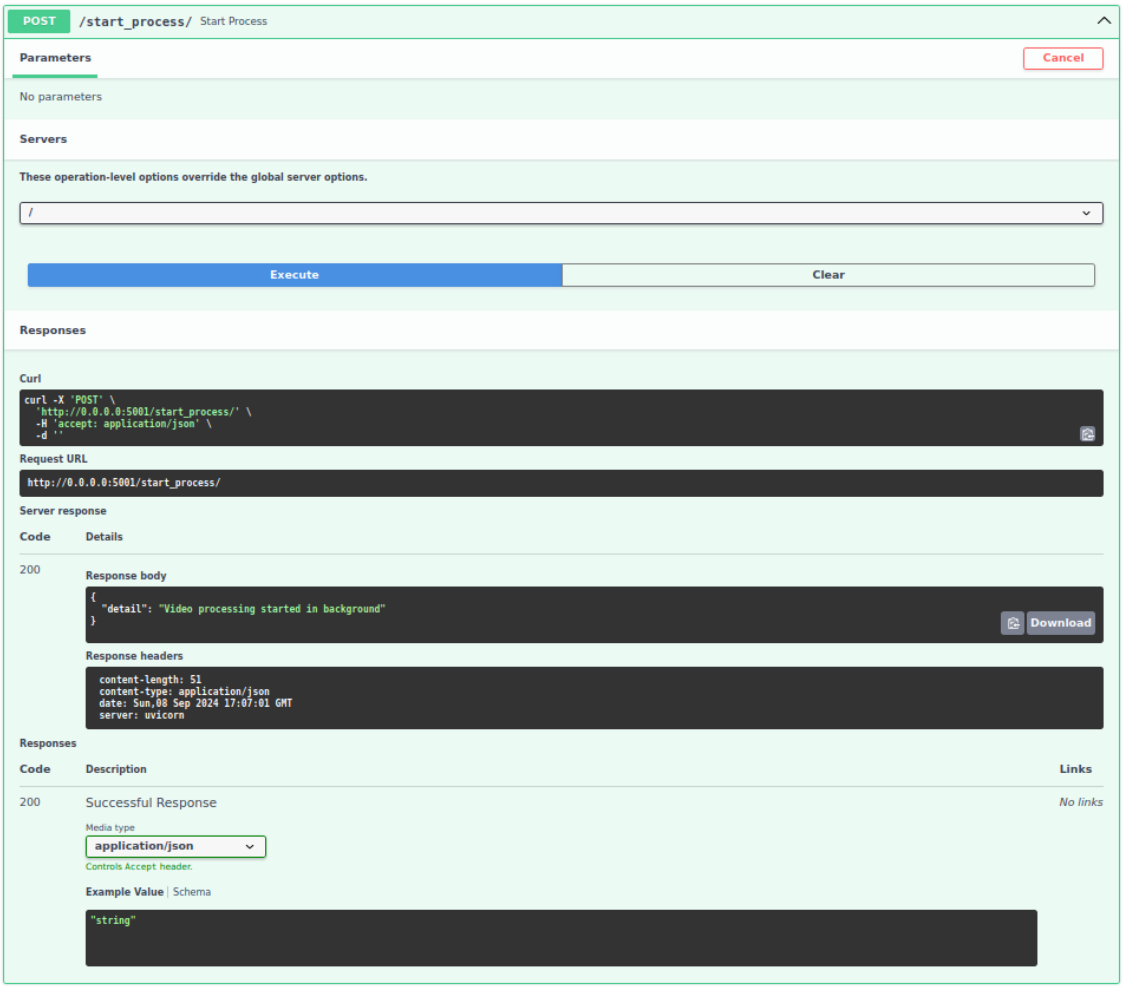


Figura 47. Endpoint de start_process en el swagger. Elaboración propia.

4.1.2.8 Queue_status

Con el fin de evitar falsos positivos y establecer una señal estable en el tiempo, también se crea una variable global consistente en un diccionario cuya clave vuelve a ser el nombre de la cámara y cuyo valor es una cola. Una cola consiste en una lista formada por los últimos valores de una señal, estando limitada en longitud. Cuando se llena, el último valor sale y el nuevo entra. El front está diseñado para hacer peticiones recursivas de este endpoint, por lo que siempre se mantiene al tanto del estado del paciente. Como se verá posteriormente, es necesario tener al menos tres señales de caída dentro de la cola para dar la señal de alarma.

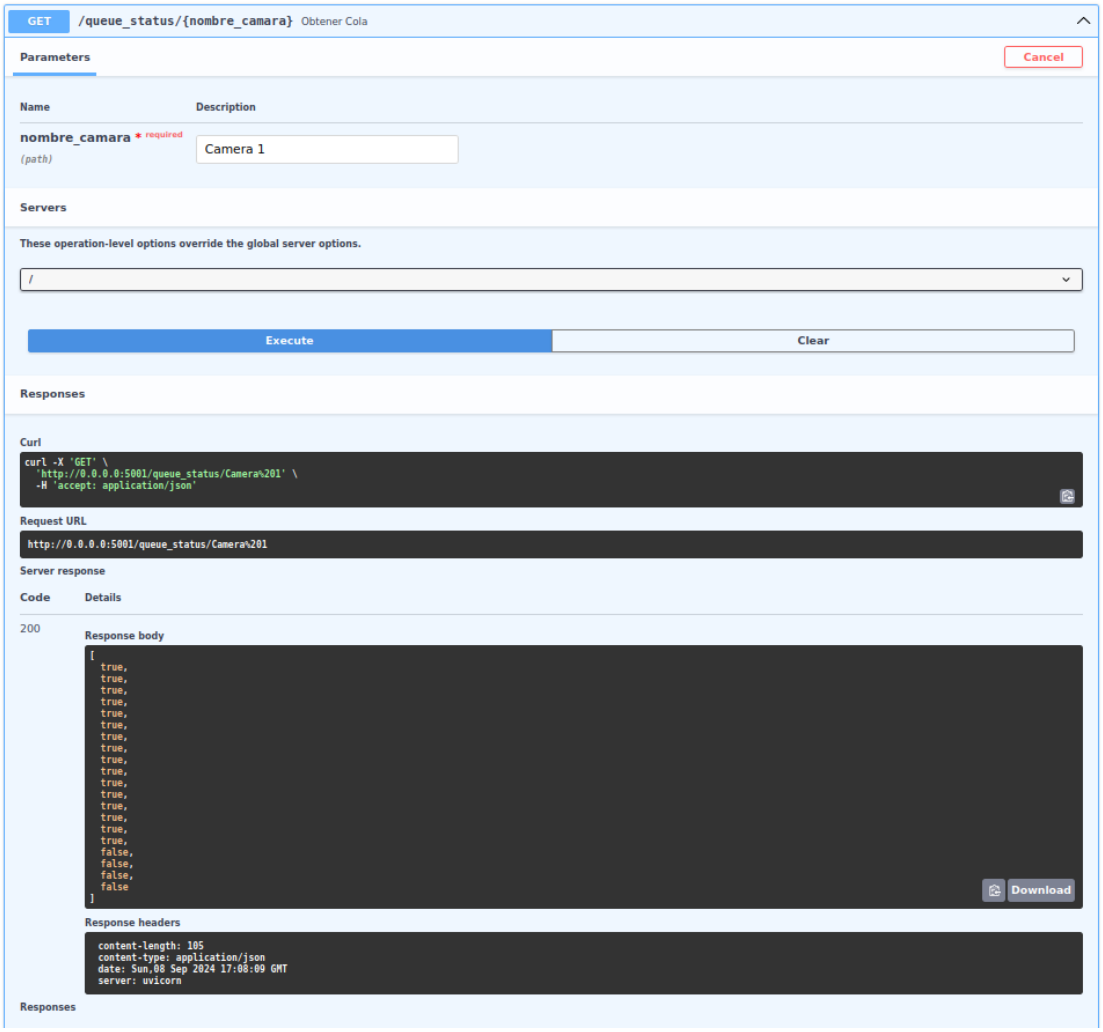


Figura 48. Endpoint de queue_status en el swagger. Elaboración propia.

4.1.3 Interacción del sistema de forma general con la API.

Como se observa, el proceso es gestionado de forma íntegra por la API y la interacción con ella es continua. Como se observa en la Figura 49, los procesos de inferencia y gestiones y las peticiones de frames y estado de cola son constantes. Esta dinámica garantiza un proceso confiable y actualizado de manera continua para que el operario tenga una información inmediata del proceso.

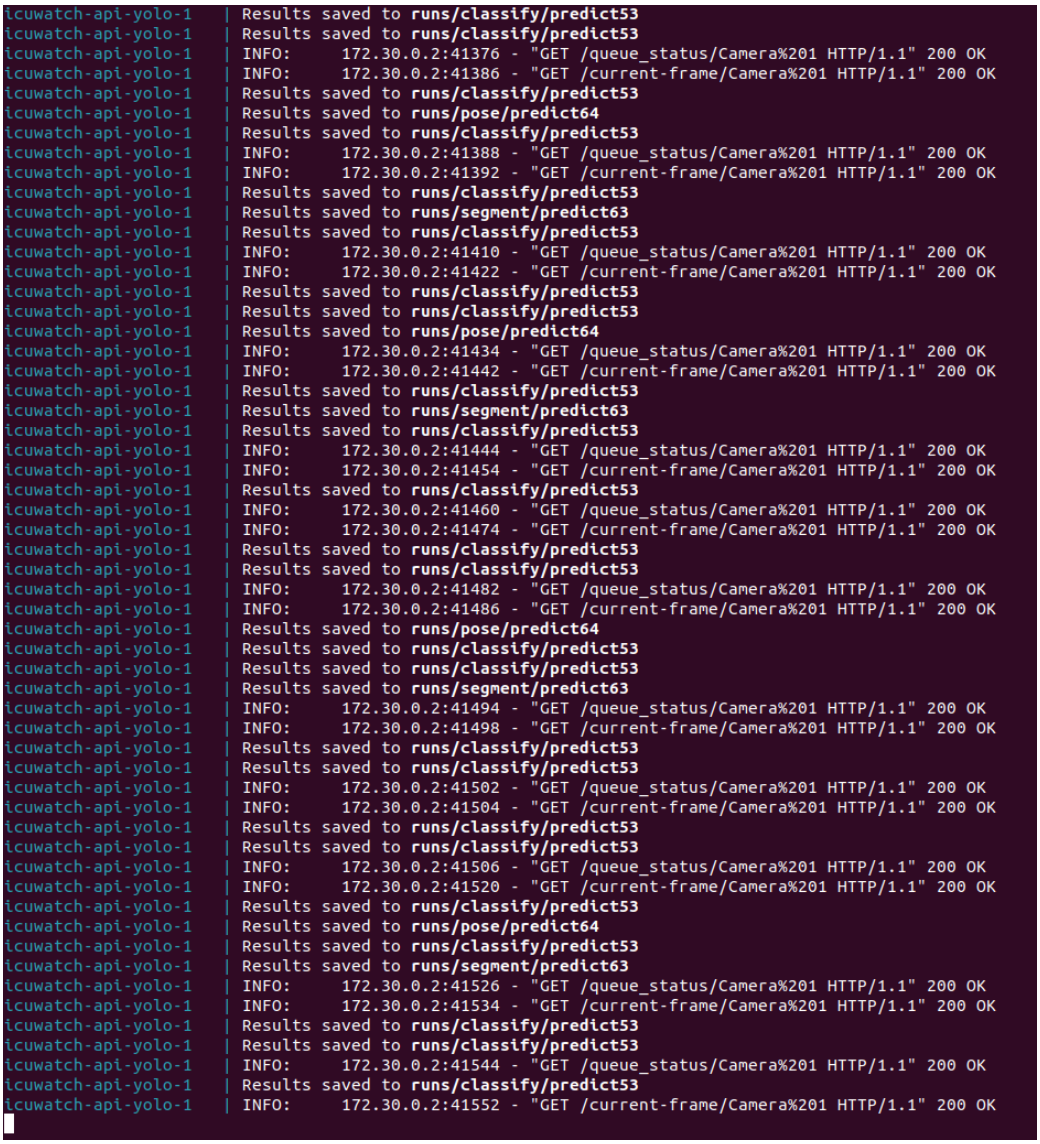


Figura 49. Vista habitual de la terminal durante el proceso. Elaboración propia.

4.1.4 Streamlit. Interfaz para la visualización de alertas.

4.1.4.1 Estructura general.

Para estructurar un front-end en forma de aplicación web se optó por el uso de Streamlit, el cual ofrece un framework en Python para la generación de interfaces gráficas y ofrece un despliegue rápido, a la par que es muy utilizado las aplicaciones de machine learning.

En el caso de estudio, se configuró una interfaz simple, con un visor y con la colocación del logo del proyecto en él. Se hizo una combinación antagónica de colores naranja y azul para la imagen del proyecto.

Para la visualización de alertas, se aprovechó las funcionalidades que ofrece Streamlit. En este sentido, al ser una aplicación web y estar estructurada en Python las posibilidades de personalización que ofrecen son limitadas. Sin embargo, es posible introducirle un formato de estilo y diseño web mediante un CSS.

Gracias a esta funcionalidad, se le pudo dotar de un diseño más profesional a la interfaz, y se creó un marco de visualización de alertas, que se mostraba en verde cuando la situación no presentaba alertas, y que parpadeaba en rojo intenso cuando la señal de alerta se activaba.

En la Figura 50 y Figura 51 se puede observar el diseño de la interfaz, así como la visualización en los escenarios de alerta y no alerta.

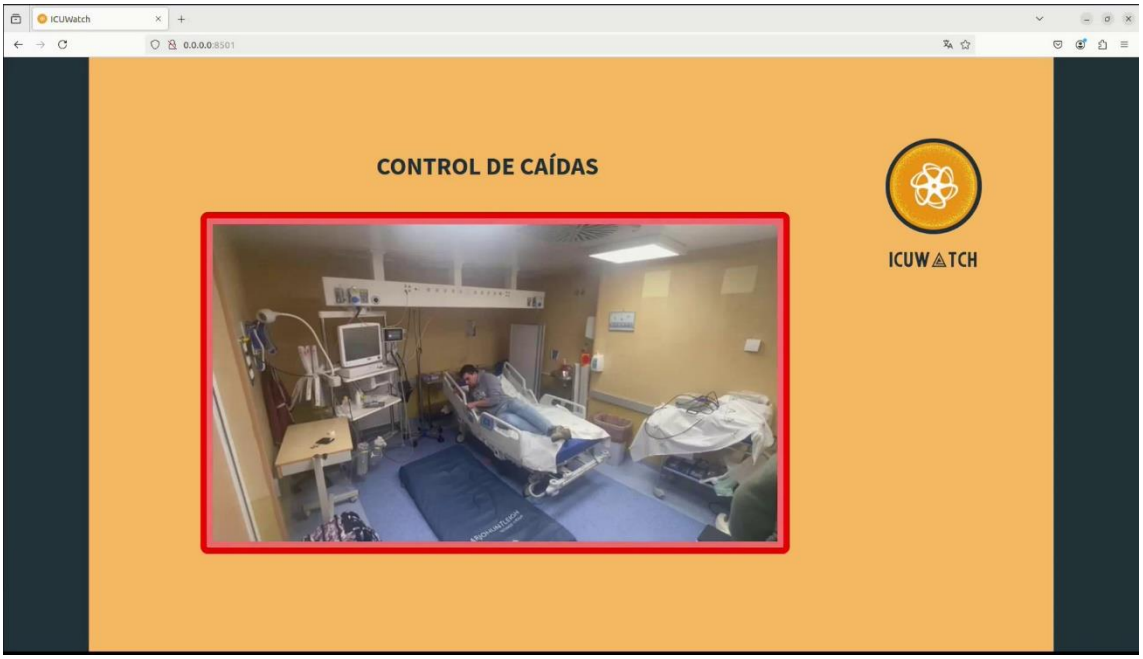


Figura 50. Interfaz gráfica con señal de caída. Elaboración propia.

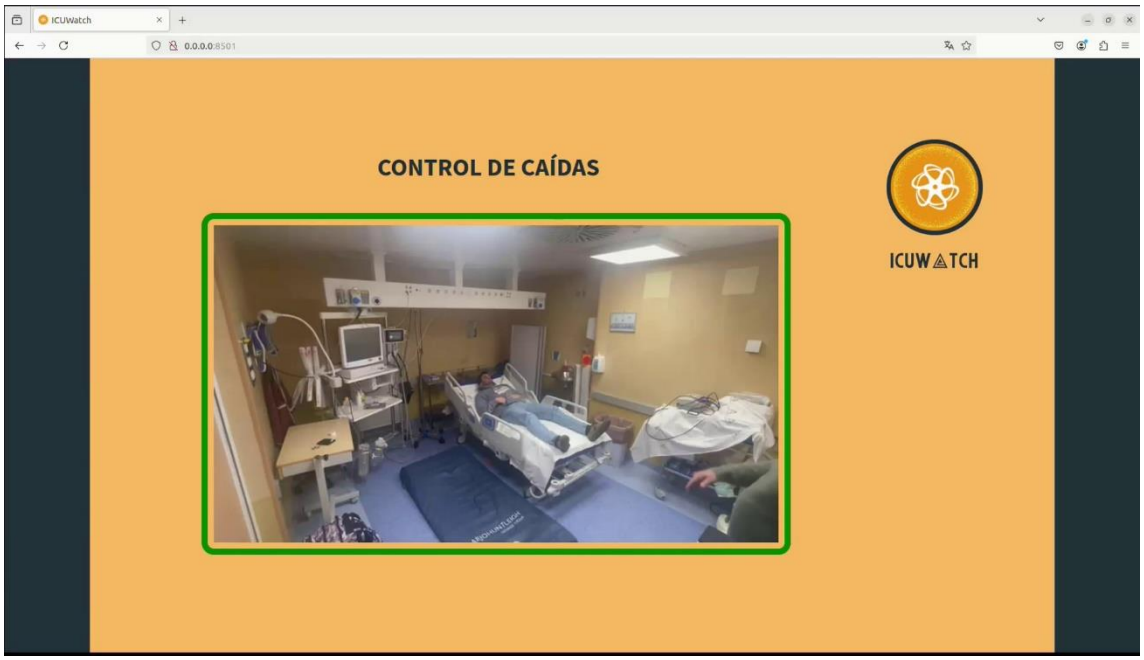


Figura 51. Interfaz gráfica en estado normal. Elaboración propia.

4.1.4.2 Gestión de alertas en la pantalla principal del técnico (sanitario).

Este servicio también está diseñado para admitir más cámaras, las cuales se irían disponiendo en forma de cuadrícula en la zona central de visualización. Así, la parte final del código de Streamlit comprende una iteración recursiva con una petición a la API de frame y cola por cada cámara presente.

Así, a la recepción la imagen, se convierte a bytes y se codifica en Base64, que es la forma correcta de mostrarlas en una aplicación web. En cuanto a la cola, se analiza y si contiene más de tres señales de caída se activa la señal de alarma.

En cuanto a la gestión de la alarma, al no repercutir de forma alguna en ningún sistema presente en planta, se mantiene activa hasta que el paciente vuelva a su posición normal. Se planteó la posibilidad de añadir botones u otros elementos de interacción con el software, pero se consideró innecesario.

4.1.5 Unificación de partes, Dockerización y funcionamiento general del sistema.

4.1.5.1 Estructura general.

Docker constituye una plataforma que permite la creación de contenedores para el desarrollo y distribución de aplicaciones. Un contenedor se entiende como una unidad cerrada, donde es posible instalar todos los requerimientos del sistema y ejecutar desde ahí la aplicación.

Este enfoque ofrece un gran número de ventajas, pero la principal es que asegura el funcionamiento del sistema independientemente del entorno donde se ejecute. Así, se puede desplegar de forma rápida y eficiente en un gran número de terminales, y permite la reproducibilidad sencilla del sistema.

En la misma línea, Docker Compose permite orquestar varios contenedores diferentes para componer una aplicación completa. Así se puede establecer un sistema de microservicios basado en la reproducibilidad de Docker.

En estos sistemas, todo gira alrededor del archivo *docker-compose.yml*, donde se especifican los servicios que se van a implementar, así como otros requerimientos, entre los que se incluyen los volúmenes, redes y puertos compartidos con el sistema huésped.

4.1.5.2 Dockerización del sistema.

La dockerización del sistema es un paso vital para el desarrollo de un proyecto serio y exportable a terminales en planta. En este sentido, es importante que se realice es las fases iniciales del proyecto, ya que al hacerlo se presentan ciertas restricciones, como los volúmenes o puertos a los que puedes acceder, y es mejor desarrollar con estas limitaciones ya presentes.

En la Figura 52 y la Figura 53 se observa cómo queda estructurado el sistema dentro del Docker Compose, con la configuración de tres microservicios. En este sentido, al front-end se le dan permisos para acceder al puerto 8501 y a la API al 5001.

De igual relevancia es la configuración del reinicio automático de contenedores. Así, si ocurre algún error en el código o existe un reinicio del sistema, los contenedores se pondrán en marcha de forma instantánea y autónoma.

Además, se incluyen los volúmenes compartidos entre el contenedor y el huésped, para tener una interacción eficaz con el sistema.

```yaml
version: '3'

services:
  streamlit:
    build:
      context: . # Directorio donde se encuentra Dockerfile_main
      dockerfile: Dockerfile # Nombre de tu Dockerfile
    healthcheck:
      disable: true
    restart: unless-stopped
    tty: true
    environment:
      - LOGLEVEL=debug
    ports:
      - "8501:8501"
    volumes:
      - ./src/:/src/
      - ./files/:/files/
      - ./temp:/temp
      - ./cfgs/:/cfgs/
    # network_mode: "host"
    entrypoint: [ "streamlit", "run", "/src/front.py", "--server.address=0.0.0.0" ]

  main:
    build:
      context: .
    healthcheck:
      disable: true
    restart: unless-stopped
    stdin_open: true
    environment:
      - INDEX=1
      - LOGLEVEL=DEBUG
      - PYTHONUNBUFFERED=0
      - DISPLAY=unix$DISPLAY
      - QT_X11_NO_MITSHM=1
      - PLC_CONNECTION_TYPE=simulated
    volumes:
      - ./files/:/files/ # cfgs
      - ./src/:/src/ # source code
      - ./weights/:/weights/ # files
      - ./cfgs/:/cfgs/
      - ./temp/:/temp/
    privileged: true
    network_mode: "host"  # pycam ethernet
    entrypoint: [ "python", "-u", "/src/main.py"]
```

Figura 52. Servicios de main y Streamlit en el docker-compose.yml. Elaboración propia.

```yaml
api-yolo:
  build:
    context: .
    dockerfile: Dockerfile
  deploy:
    resources:
      reservations:
        devices:
          - capabilities:
            - gpu
  ports:
    - "5001:5001"
  restart: unless-stopped
  healthcheck:
    test: ["CMD", "curl", "-f", "http://api-yolo:5001/healthcheck"]
    interval: 10s
    timeout: 10s
    retries: 5
    start_period: 10s
  volumes:
    - ./files/:/files/ # cfgs
    - ./src/:/src/ # source code
    - ./weights/:/weights/ # files
    - ./cfgs/:/cfgs/
    - ./temp/:/temp/
  environment:
    - PYTORCH_VRAM_LIMIT=0.7
    - LOGLEVEL=debug
    - NVIDIA_VISIBLE_DEVICES=all
    - NVIDIA_DRIVER_CAPABILITIES=compute,utility
  stdin_open: true
  logging:
    driver: "json-file"
    options:
      max-size: "20m"
      max-file: "10"
  command: ["uvicorn", "src.api_yolo:app", "--host", "0.0.0.0", "--port", "5001"]
```

Figura 53. Microservicio de la API en el docker-compose.yml. Elaboración propia.

Por último, cabe destacar una particularidad de este sistema. Es habitual que los contenedores de Docker se levanten a partir de una imagen, que no es más que un registro inmutable de código. En este caso, nos encontramos en el paso previo a esa dinámica, por ello debemos definir un sistema base en el Dockerfile, como se ve en la Figura 53, en este caso Python 3.10, en el que se deben instalar todas las dependencias necesarias, presentes en la Figura 54. En el Dockerfile también es recomendable dejar habilitados los puertos para las APIs.

Dockerfile > ...

```dockerfile
FROM python:3.10.12

COPY requirements.txt requirements.txt

RUN apt-get update
RUN apt install -y libgl1-mesa-glx
RUN pip install --trusted-host pypi.python.org -r requirements.txt

CMD ["uvicorn", "src.api_yolo:app", "--host", "0.0.0.0", "--port", "5001"]
```

Figura 54. Dockerfile. Elaboración propia.

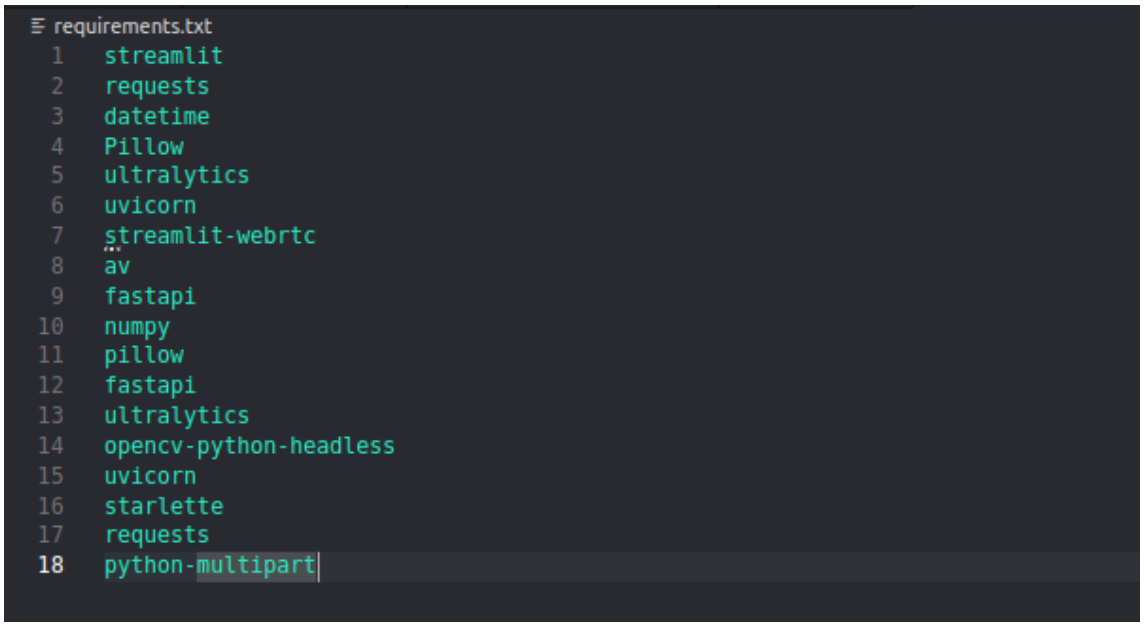


Figura 55. Requirements.txt. Elaboración propia.

Esta es la razón por la que en la Figura 52 y la Figura 53 se definan los denominados *entrypoints*, como archivos a partir de los que ejecutar el contenedor.

4.1.5.3 Funcionamiento. Gestión del main.

Como se ha comentado con anterioridad, la función de los servicios propios de lo que comúnmente se denomina main, corresponde en este caso a la API. El main de este sistema se encarga de forma principal de asegurar que todos los servicios se levanten correctamente.

Así, al iniciarse el sistema, el main espera 10 segundos a que los contenedores se levanten. Esta función es vital, ya que si no las peticiones entre contenedores empezarían a darse desde el principio y esto incurriría en fallos del sistema. Un fallo del sistema provoca el reinicio de dicho contenedor, por lo que tendríamos un reinicio continuo.

Tras este tiempo de espera carga los modelos, lo cual es una acción casi instantánea y no requiere acciones anexas ni tiempos de espera, ya que no imposibilita otras acciones.

Posteriormente, se hace una llamada al endpoint de *start_video*, y se introduce nuevamente un tiempo de espera de 10 segundos, ya que si el número de cámaras es elevado puede demorarse un poco.

Este tiempo de espera es necesario, ya que la siguiente llamada a la API es la del endpoint *process_cameras*, y si no se esperara pasaría como el caso anterior, ya que todavía no habría video que iniciar. Sin embargo, existen contingencias dentro de la API que prevén esta posición.

Se hace una nueva espera de cinco segundos y se llama al endpoint de s*tart_process*, que comienza con el proceso propiamente dicho y las llamadas a la cola desde el front comienzan a darse.

Como se puede observar, el main solo actúa como inicializador controlado del sistema, y su actuación no se extiende en tareas operativas.

4.1.5.4 Evaluación paramétrica del sistema y optimización (tiempos de ciclo etc.)

En todos los sistemas es muy relevante realizar un estudio de parametrización del sistema, así como de aprovechamiento de recursos. En este sentido, lo más relevante es ser consciente del equipo utilizado en las pruebas, especialmente a nivel de CPU.

Esto se debe principalmente a que la GPU y la RAM son especificaciones que, si bien tienen baremos de calidad, es cuantificable el requerimiento superior que sería necesario. Por ejemplo, si se tiene tres modelos de visión artificial, y se sabe que cada uno ocupa 2 GB de memoria gráfica, como mínimo será necesaria una GPU de 6 GB. En el caso de las CPU es diferente porque un mayor número de núcleos no sigue esa relación tan directa. Así, los datos del equipo de prueba se pueden observar en la Figura 56 y la Figura 57.

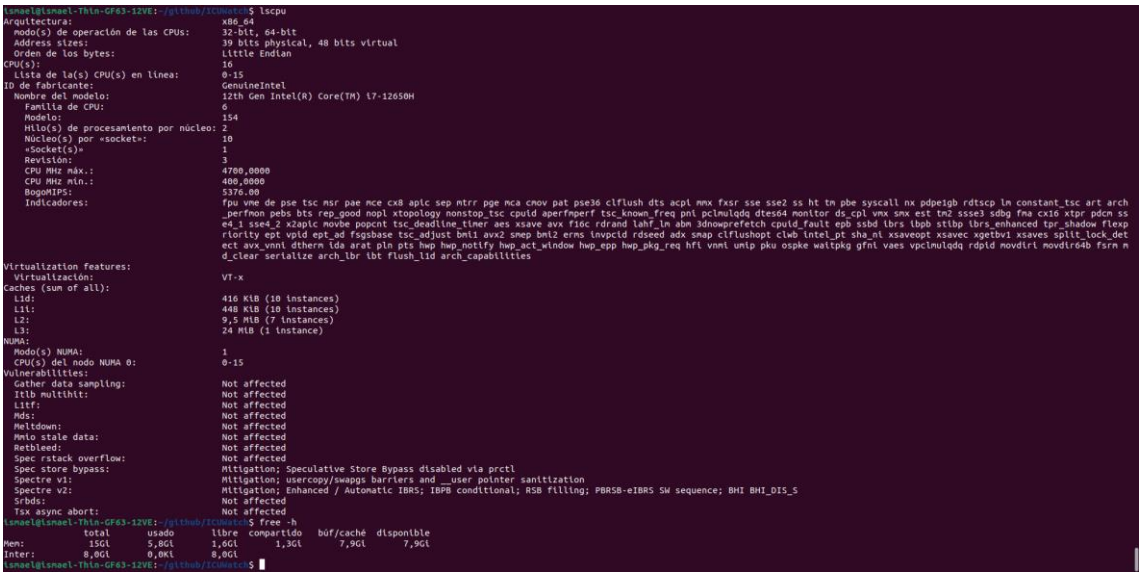


Figura 56. Especificaciones del terminal de validación (CPU Y RAM). Elaboración propia.

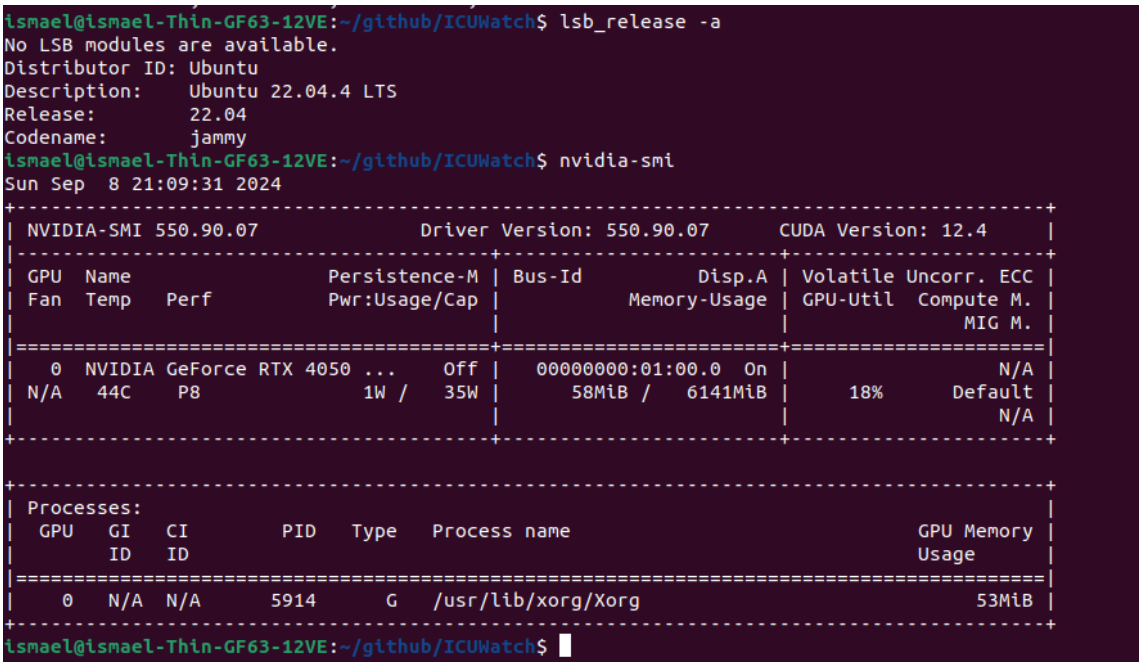


Figura 57. Especificaciones del terminal de validación (GPU y SO). Elaboración propia.

Así, con el sistema plenamente operativo con el uso de una cámara, los datos de uso del terminal relativos al uso de CPU y RAM se pueden observar en la Figura 58, y en lo relativo al uso de la tarjeta gráfica, se pueden observar en la Figura 59.

Figura 58. Rendimiento del equipo bajo proceso (CPU y RAM). Elaboración propia.

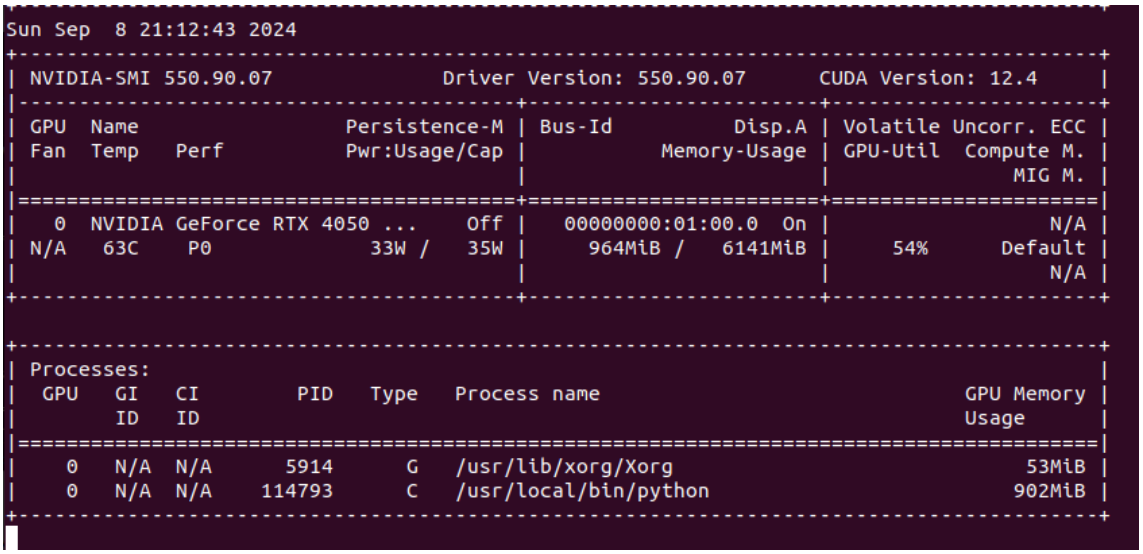


Figura 59. Rendimiento del equipo bajo proceso (GPU). Elaboración propia.

Estos datos son muy buenos, sobre todo con relación al uso de la tarjeta gráfica. Los datos de la CPU y la RAM son excelentes para usar una sola cámara, pero enfrenta serios problemas cuando aumentan, ya que el número de hilos genera altos requerimientos para la CPU.

En relación con el número de frames por segundo, el sistema es capaz de procesar el video a plena capacidad, que en el caso simulado incurren en unos diez frames por segundo.

Estos resultados reflejan la situación de rendimiento tras diversos ajustes, como los realizados a través del cambio en cómo cargar los modelos. Que tres modelos de este tipo, siendo uno de ellos de segmentación y otro de ellos un L solo ocupen 1 GB es realmente muy buen dato y anima a avanzar en esa dirección en proyectos futuros.

Asimismo, en relación con el número de frames, durante el desarrollo hubo problemas con la gestión de los mismos, así como saturar el requerimiento del sistema al ir demasiado rápido. Mediante la petición a la API tanto de imagen como de lista se elimina esta posibilidad, ya que, si la solicitud se hace demasiado rápido, el front-end simplemente recibirá el mismo resultado, el cual publicará.

4.2 Manuales

De cara al consumidor final no aplican manuales, ya que el operario no debe interaccionar activamente con el sistema.

4.3 Pruebas

4.3.1 Evaluación del sistema y validación.

Para la validación se estudiaron varias situaciones críticas en momentos previos a caídas. Para ello, se hizo un análisis no solo de lo reflejado por el front-end en el tramo final, sino también en etapas intermedias. Así, se pueden observar diferentes situaciones, con planos anteriores y posteriores a los momentos críticos.

Cabe destacar que los frames que se muestran no son consecutivos, ya que si fuera ese el caso no daría lugar a la lista a llenarse, por lo que son secuencias con frames separados entre sí.

Así, en la Figura 60 se puede ver de forma precisa que la lógica relacionada con los Keypoints es correcta, ya que en las dos primeras debería dar situación normal, al no haber suficientes Keypoints dentro de zona.

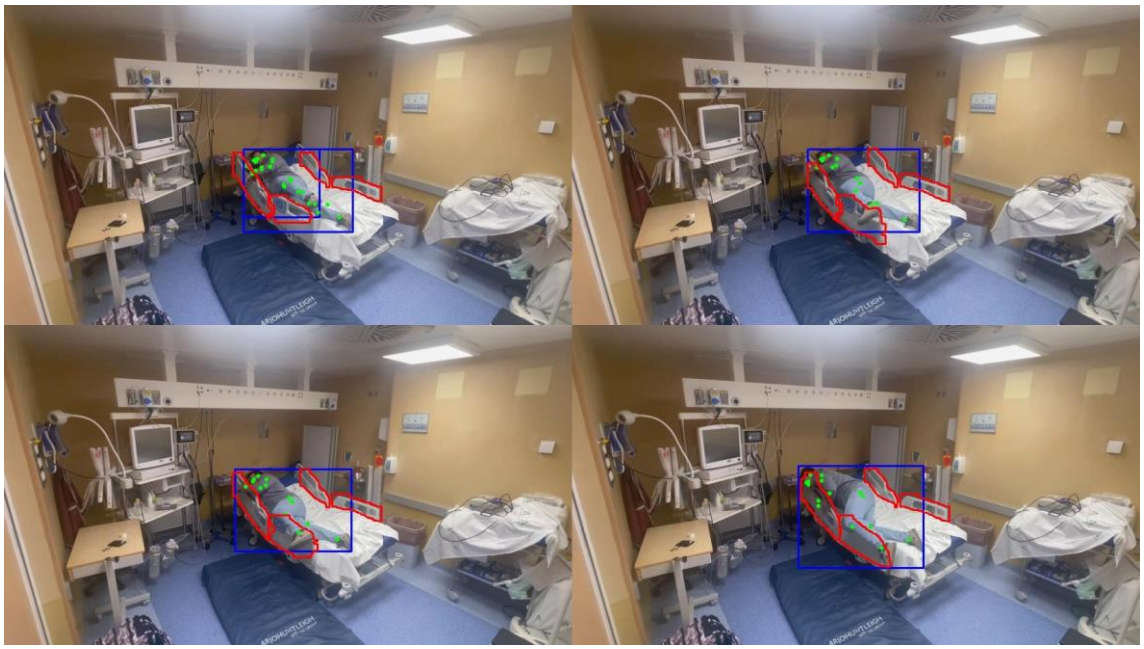


Figura 60. Detecciones de modelos para validación 1. Elaboración propia.

Esta lógica se ve refrendada en la Figura 61, que da situaciones normales en las primeras dos fotografías, y da caída y alerta en las siguientes dos.

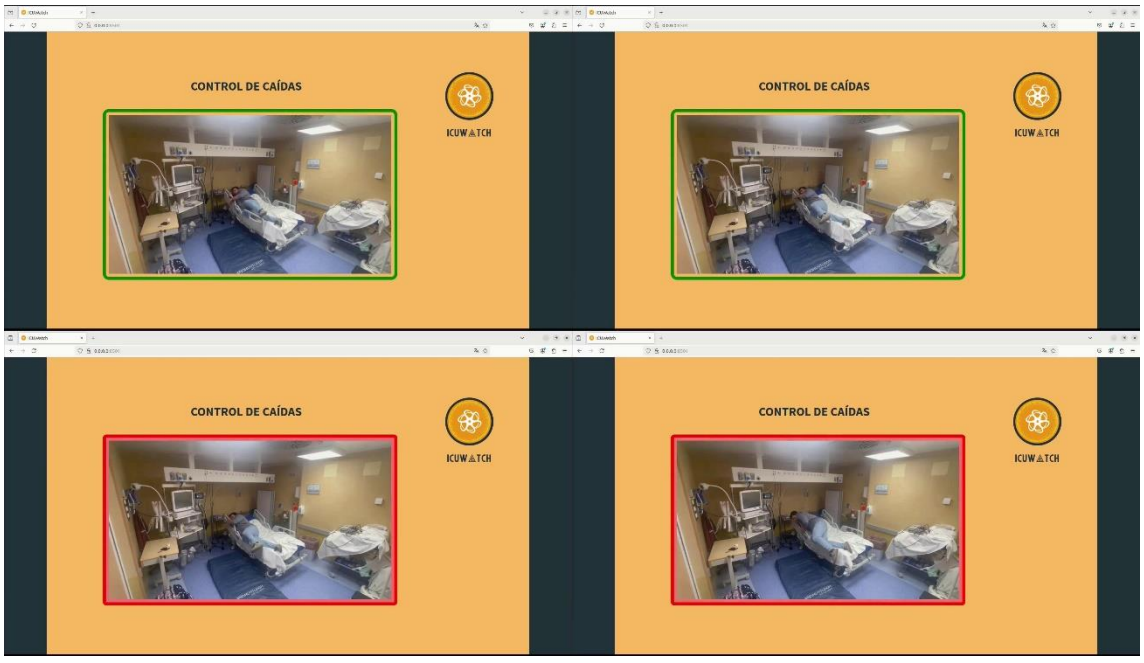


Figura 61. Visión de interfaz para validación 1. Elaboración propia.

Asimismo, se puede observar en un segundo caso. Esta situación es algo más complicada, ya que los hechos ocurren de una forma más rápida. En la Figura 62 se obtiene que en los primeros frames no hay apenas Keypoints. Posteriormente, los puntos clave seleccionados están dentro de zona y se valida la caída.

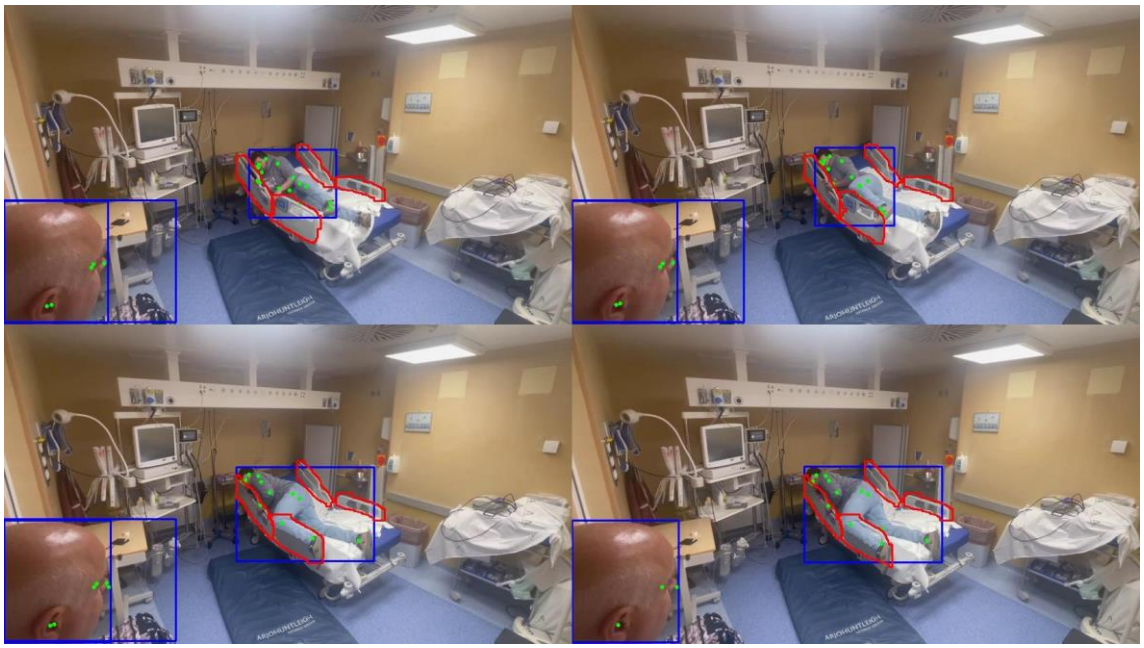


Figura 62. Detecciones de modelos para validación 2. Elaboración propia.

En la Figura 63 se corrobora que el análisis es correcto, ya que los dos primeros frames devuelven una situación normal y los dos segundos alerta por caída. Bien es cierto que la acción es más rápida y eso repercute en el tiempo que tarda en darse la señal. Se debe recordar que el paciente objetivo de este sistema está en estado de semiinconsciencia, por lo que no debería dar problemas en este sentido.

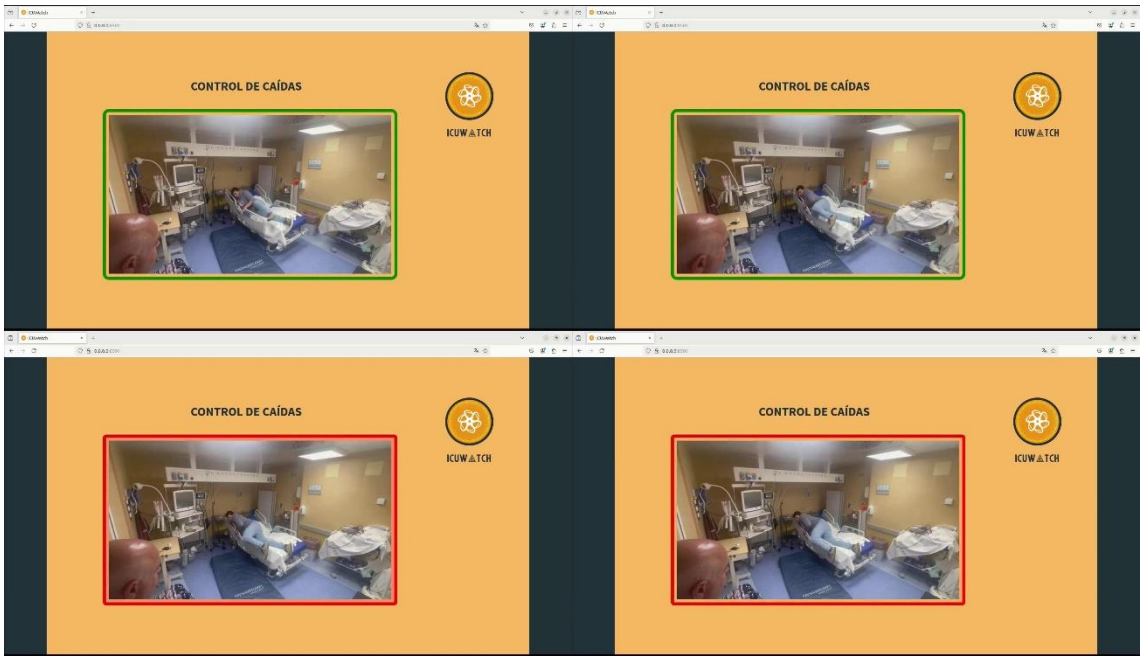


Figura 63. Visión de interfaz para validación 2. Elaboración propia.

Dentro de las comprobaciones para la validación del sistema también se debe validar que el sistema vuelve a su estado inicial cuando el paciente recupera la posición. En este sentido, en la Figura 64 el paciente viene de una posición de caída, y la cola está llena.

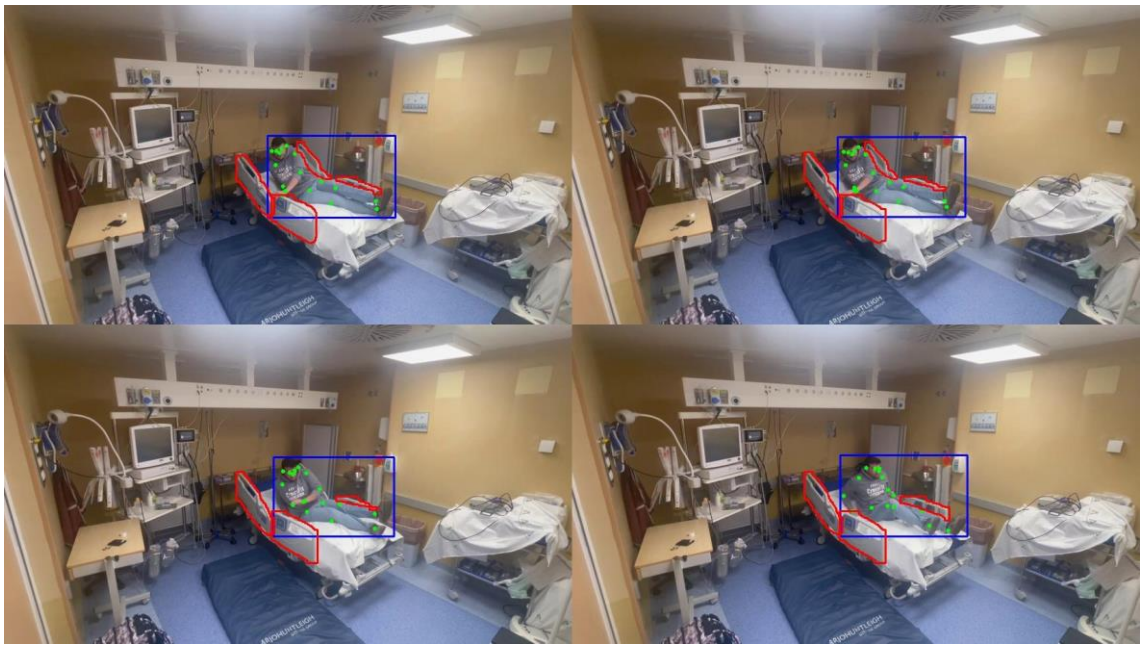


Figura 64. Detecciones de modelos para validación de recuperación. Elaboración propia.

El paciente vemos como se encuentra con los Keypoints fuera de barreras, por lo que la cola debería vaciarse. Comprobamos que esta situación se da en la Figura 65, donde al final, cuando se acomoda, la alarma de caída pasa de activa a inactiva.

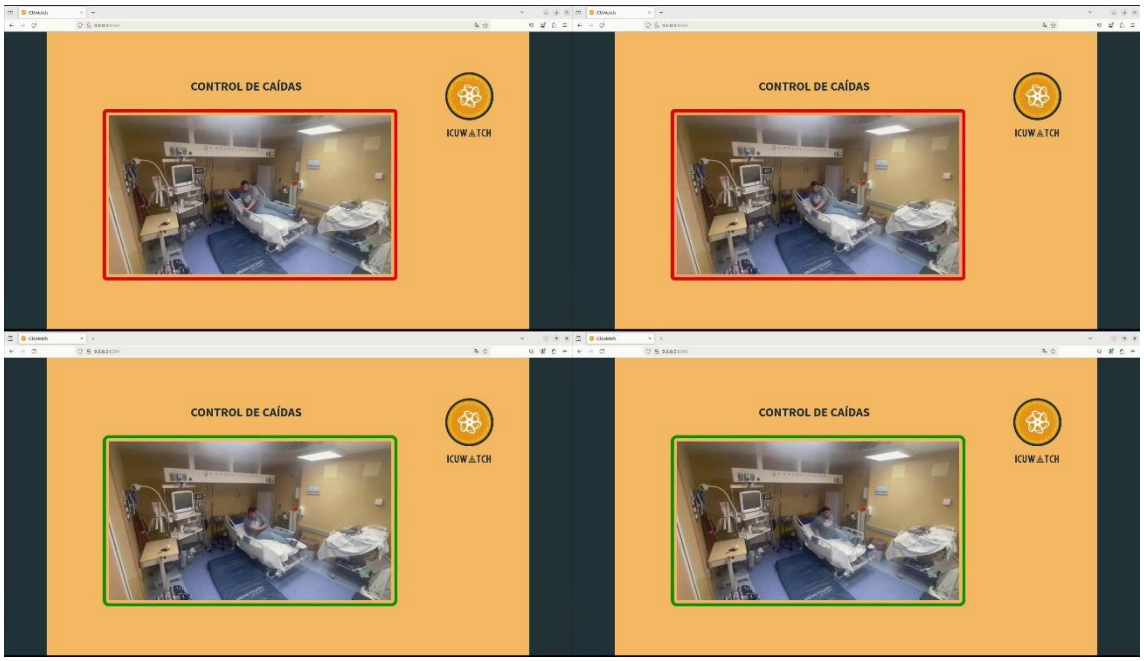


Figura 65. Visión de interfaz para validación de recuperación. Elaboración propia.

5 Conclusiones.

En el ámbito médico, las implicaciones que van a suponer el desarrollo de metodologías de esta índole en el mundo de la medicina pueden suponer una verdadera revolución en el mundo de la seguridad del paciente. Estas metodologías mejorarán tanto la vida de los pacientes, como la calidad del trabajo de los sanitarios.

Durante la realización de este proyecto, se pudieron recopilar innumerables noticias relacionadas con esta índole, y es que especialistas en el sector comentan que es un problema de primer nivel, y que vías de investigación como esta evitarán muchos daños innecesarios, procedimientos adicionales, estancias hospitalarias prolongadas e incluso costes, viendo el beneficio de todas las partes implicadas.

En un ámbito más ingenieril, cabe destacar el aprendizaje derivado de los retos a los que me he enfrentado en la realización de este proyecto. El primero de ellos fue la recolección de datos de carácter médico, lo cual fue una tarea muy difícil, y que me hubiera resultado mucho más complicada si no hubiese contado con apoyo especializado.

Asimismo, los problemas con la gestión de recursos y la carga de modelos para la viabilidad presentaron un reto cuya resolución a nivel profesional abre un nuevo abanico en cuanto a los recursos necesarios para desplegar un sistema de estas características, así como incentiva la investigación a la disminución de requerimiento de recursos en otras áreas.

En el ámbito del proyecto en sí, queda la perspectiva de la expansión del sistema a una proyección de más cámaras. El proyecto tomaría una dimensión comercial diferente como herramienta de ayuda y atención a un servicio hospitalario completo, el cual se estructura entre 6 y 10 camas, al menos.

En este sentido, la perspectiva no es muy lejana y el código, así como el framework estarían preparados para esa mejora. Para ello, sería muy importante una mejora en las especificaciones del terminal.

La mejora de estas especificaciones pasa por trasladar el sistema a un terminal con más potencia de procesamiento, pero puede que la vía de intentar buscar frameworks y otras metodologías que permitan una gestión más eficiente de los recursos disponibles siga dando tan buenos frutos como hasta ahora. En general, las perspectivas para una ampliación del proyecto son buenas y los datos arrojados han sido muy positivos.

6 Referencias.

[1] OMS, «Plan de Acción Mundial para la Seguridad del Paciente 2021-2030».

[2] M. Boot, J. Allison, J. Maguire, y G. O'Driscoll, «QI initiative to reduce the number of inpatient falls in an acute hospital Trust», *BMJ Open Qual*, vol. 12, n.º 1, feb. 2023, doi: 10.1136/bmjoq-2022-002102.

[3] «La Junta pagará más de 45.000 euros a los hijos de una mujer que murió en una residencia tras una caída de la cama». Accedido: 20 de agosto de 2024. [En línea]. Disponible en: https://www.diariodesevilla.es/juzgado_de_guardia/actualidad/junta-pagara-45000-euros-muerte-residencia-caida_0_2002217692.html

[4] T. Health y H. Services, «Evidence-Based Best Practices: Physical Restraints».

[5] «Prevención de caídas en pacientes hospitalizados - Manuales Clínicos». Accedido: 30 de agosto de 2024. [En línea]. Disponible en: https://manualclinico.hospitaluvrocio.es/procedimientos-generales-de-enfermeria/cuidados-basicos/prevencion-de-caidas-en-pacientes-hospitalizados/

[6] «[1511.08458] An Introduction to Convolutional Neural Networks». Accedido: 30 de agosto de 2024. [En línea]. Disponible en: https://ar5iv.labs.arxiv.org/html/1511.08458

[7] M. D. Zeiler y R. Fergus, «Stochastic pooling for regularization of deep convolutional neural networks», *1st International Conference on Learning Representations, ICLR 2013 - Conference Track Proceedings*, 2013.

[8] N. Scholz, J. Moll, M. Mälzer, K. Nagovitsyn, y V. Krozer, «Random bounce algorithm: real-time image processing for the detection of bats and birds: Algorithm description with application examples from a laboratory flight tunnel and a field test at an onshore wind energy plant», *Signal Image Video Process*, vol. 10, n.º 8, pp. 1449-1456, nov. 2016, doi: 10.1007/s11760-016-0951-0.

[9] «Detección de objetos I: YOLO • Un artículo de LMO». Accedido: 30 de agosto de 2024. [En línea]. Disponible en: https://lamaquinaoraculo.com/deep-learning/deteccion-de-objetos/

[10] «YOLOv7 Pose vs MediaPipe in Human Pose Estimation». Accedido: 30 de agosto de 2024. [En línea]. Disponible en: https://learnopencv.com/yolov7-pose-vs-mediapipe-in-human-pose-estimation/

[11] «imgaug — imgaug 0.4.0 documentation». Accedido: 27 de agosto de 2024. [En línea]. Disponible en: https://imgaug.readthedocs.io/en/latest/

[12] A. Saxena *et al.*, «A review of clustering techniques and developments», *Neurocomputing*, vol. 267, pp. 664-681, dic. 2017, doi: 10.1016/j.neucom.2017.06.053.