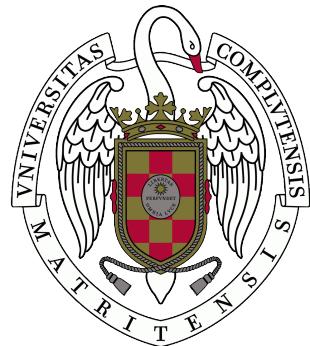

**Reconocimiento y análisis de personas en
imágenes mediante técnicas de aprendizaje
profundo**

**Recognition and analysis of people in images
using deep learning techniques**



**Trabajo de Fin de Grado
Curso 2024–2025**

Autores

Ignacio García Fernández
Jorge Serrano Velázquez
Jorge Torres Ruiz

Director

Gonzalo Pajares Martinsanz

Colaboradora Externa

Clara Isabel López González

Grado en Ingeniería Informática e Ingeniería del Software

Facultad de Informática

Universidad Complutense de Madrid

Reconocimiento y análisis de personas en imágenes mediante técnicas de aprendizaje profundo

Recognition and analysis of people in images using deep learning techniques

Trabajo de Fin de Grado en Ingeniería Informática e Ingeniería de Software

Autores

Ignacio García Fernández
Jorge Serrano Velázquez
Jorge Torres Ruiz

Director

Gonzalo Pajares Martinsanz

Colaboradora Externa
Clara Isabel López González

Convocatoria: Junio 2025

Grado en Ingeniería Informática e Ingeniería del Software
Facultad de Informática
Universidad Complutense de Madrid

26 de mayo de 2025

Dedicatoria

Ignacio García Fernández

A mi madre, por todos sus sacrificios y por apoyarme tanto en los buenos como en los malos momentos durante todos estos años; por estar presente en cada una de mis decisiones e inculcarme los valores que me han permitido alcanzar mis objetivos.

A mi padre, por sus esfuerzos, sus consejos, y por estar ahí siempre que lo he necesitado, apoyándome de manera incondicional a lo largo de todo este tiempo.

A mis abuelos, de los cuales he aprendido que todo lo que merece la pena requiere esfuerzo, trabajo y dedicación; que nada se regala, y que todo trabajo debe ir acompañado de humildad, independientemente de lo que se persiga.

Por último, a mi segunda familia: mis amigos, estoy profundamente agradecido de contar con un grupo de personas tan cercanas, que durante este tiempo me ha permitido compartir tanto los malos como los buenos momentos. Gracias a ellos sigo creciendo como persona, y es por eso, en un momento como este, quiero acordarme de vosotros: David, Dani F, Jorge S, Lucas, Javi, Fran, Jota, Ruth, Fabio y Rober.

Jorge Serrano Velázquez

A mis padres, por su apoyo incondicional en los momentos difíciles y por haber estado a mi lado durante todo este camino, no solo en la carrera, sino en la vida.

A mi hermana, que siempre me ha ayudado con todo lo que ha podido, estando cerca y estando a millas de distancia.

A mi abuela, que siempre ha estado pendiente de ponerme velas en los exámenes.

A todos mis amigos del “Sete’s Beach Club”, por ponerme una cervecita en la mano cuando había que desconectar y animarme a acabar la carrera al año.

A mi pareja Andrea, por estar siempre a mi lado, animándome tanto en la carrera y el TFG como en todo lo demás, gracias por tu apoyo incondicional, dentro y fuera de los estudios.

A mis compañeros de carrera, por animarme, acompañarme y hacer este camino mucho más llevadero entre risas, apoyo y complicidad. Especialmente a Yoha que aunque ya no esté con nosotros, siempre quedará una parte suya en nuestras vidas.

Jorge Torres Ruiz

A mis padres y hermanos, por apoyarme en todas las decisiones que he tomado a lo largo de mi vida. Gracias por estar siempre a mi lado, impulsándome a alcanzar mis metas y por haberme ayudado a convertirme en la persona que soy.

A mi grupo de amigos, que desde ese Camino de Santiago habéis sido una parte indispensable de mi vida, gracias por tantos buenos momentos, todos los viajes realizados y los que nos quedan por vivir. Llenáis mi vida de una inmensa alegría y locura. Gracias por ser como sois: Miguel, Carlos, Roberto, Manu, Bea, Helena, Jimena, Sofi, Iria, Marina y Paula.

Por último, no quiero olvidarme de todas las personas que componen la Parroquia Santa Paula, mi segunda familia. Por tantos veranos que fueron más que vacaciones, llenos de momentos inolvidables, el campamento, el Camino de Santiago, Roquetas o la JMJ.

Agradecimientos

Agradecemos al grupo ISCAR (Ingeniería de Sistemas, Control, Automatización y Robótica) de la Facultad de Informática por el apoyo prestado, especialmente al profesor D. José Luis Risco Martín por la administración, soporte y apoyo prestado en la ejecución de los procesos en el servidor puesto a nuestra disposición.

Queremos expresar nuestro sincero agradecimiento al profesor Gonzalo Pajares Martínsanz, director de este trabajo, por su constante disposición a guiarnos, por sus consejos acertados y por confiar en nuestras capacidades durante todo el proceso. Su experiencia y claridad han sido fundamentales.

También agradecemos profundamente a Clara Isabel López González por su ayuda y colaboración a la hora de usar el servidor.

Resumen

Reconocimiento y análisis de personas en imágenes mediante técnicas de aprendizaje profundo

La realización de este trabajo presenta dos objetivos fundamentales. En primer lugar, se lleva a cabo una investigación acerca del área de los métodos reconocedores de objetos en imágenes. Dicha investigación consiste en comprender los mecanismos subyacentes a estos métodos, tales como las Redes Neuronales Convolucionales (RNC) y las operaciones matemáticas que las complementan, por ejemplo, el agrupamiento o las funciones de activación. Esta tarea proporciona los conocimientos necesarios para proceder al entrenamiento de determinados modelos de detección. Para este proyecto se ha determinado que la detección será de personas en imágenes aéreas tomadas por drones en entornos naturales. Como puede observarse, el hecho de que los modelos funcionen correctamente en este ámbito puede ser de gran ayuda en diversas actividades, tales como la localización de personas desaparecidas, la prevención de accidentes o la monitorización de actividades humanas en entornos naturales.

Los modelos estudiados, basados en RNC, forman parte de lo que se denomina modelos de un solo estado, concretamente, la familia *YOLO* (*You Only Look Once*). Se lleva a cabo una fase exploratoria de las diferentes variantes, observándose cómo se comportan dichos modelos sobre las imágenes seleccionadas como *dataset*. Esta fase de investigación permite recabar los datos y resultados suficientes para llegar a la conclusión de cuál es la configuración que mejor permite resolver el problema abordado.

En segundo lugar, se procede al desarrollo de una aplicación que permita a usuarios externos entrenar modelos y comprobar su robustez con algunas imágenes de test. La aplicación pone a disposición del usuario una serie de configuraciones para llevar a cabo el entrenamiento. De esta forma, el usuario puede generar modelos con variaciones y, posteriormente, gracias a la funcionalidad de clasificación, comprobar cuál satisface mejor sus objetivos.

Palabras clave

Entornos naturales, Aprendizaje profundo, Reconocimiento de personas en imágenes, Redes neuronales convolucionales, YOLOv4, YOLOX

Abstract

Recognition and analysis of people in images using deep learning techniques

The accomplishment of this work presents two fundamental objectives. Firstly, research will be conducted on the area of object recognition methods in images. This research involves understanding the underlying mechanisms of these methods, such as Convolutional Neural Networks (CNNs) and the mathematical operations that complement them, such as pooling or activation functions. This task provides team members with the necessary knowledge to proceed with the training of specific detection models. For this project, it has been determined that the detection will focus on people in aerial images taken by drones in natural environments. As can be observed, having models function correctly in this context can be highly beneficial for various activities, such as locating missing persons, accident prevention, or monitoring human activities in natural environments.

The models to be used are part of what is called single-stage models, specifically from the YOLO (You Only Look Once) family. An exploratory phase of the different variants is carried out, observing how these models behave on the images selected as the dataset. This research phase allows students to gather sufficient data and results to conclude which configuration best solves the addressed problem.

Secondly, the development of an application proceeds that allows external users to train models and verify their robustness with some test images. The application provides users with a series of configurations for training. In this way, the user can generate models with variations and subsequently, through the testing functionality on images, verify which one best meets their objectives.

Keywords

Natural environments, Deep learning, Person recognition in images, Convolutional neural networks, YOLOv4, YOLOX

Índice

1. Introducción	1
1.1. Antecedentes	1
1.2. Motivación	3
1.3. Objetivos	4
1.4. Plan de trabajo	5
1.5. Alcance y limitaciones	7
1.6. Organización de la Memoria	7
1.7. Introduction	8
1.8. Background	8
1.9. Motivation	9
1.10. Objectives	10
1.11. Work Plan	11
1.12. Scope and Limitations	13
1.13. Thesis Organization	14
2. Conceptos, métodos y técnicas aplicadas	15
2.1. Introducción	15
2.2. Tensores	15
2.3. Métodos de optimización	16
2.3.1. Concepto de optimización	16
2.4. Funciones de activación	20
2.4.1. Sigmoid	20
2.4.2. ReLU	21
2.4.3. LeakyReLU	21
2.4.4. SoftMax	22
2.5. Redes neuronales convolucionales (CNN)	22
2.5.1. Operación de convolución	22
2.5.2. Agrupamiento (Pooling)	24
2.5.3. Sobreajuste	26
2.5.4. Normalización	27
2.5.5. Mean Squared Error	28
2.5.6. Cross-Entropy Loss	28
2.6. Detección de objetos en imágenes	28

2.6.1. Introducción	28
2.6.2. YOLO	32
3. Diseño y análisis de la aplicación: descripción del trabajo	39
3.1. Arquitectura	39
3.1.1. Capas de la aplicación	39
3.2. Casos de uso	40
3.3. Inicio de la aplicación	41
3.4. Fase de entrenamiento	42
3.4.1. Modelos, optimizadores e hiperparámetros	42
3.5. Fase de clasificación	44
3.5.1. Carga de imágenes	44
3.5.2. Configuración de opciones	44
3.5.3. Proceso de clasificación	45
3.5.4. Visualización de resultados	48
3.5.5. Herramientas adicionales	48
3.6. Gestión del proyecto	48
3.6.1. Metodología	48
3.6.2. User Story Map e Historias de Usuario	49
3.6.3. Work in progress	50
3.6.4. Reuniones	51
3.6.5. Hitos	52
3.6.6. Gestión de configuración del software	53
3.6.7. Riesgos	53
4. Análisis de resultados	55
4.1. Recursos	55
4.1.1. Dataset y licencia	55
4.1.2. Descripción del dataset	56
4.1.3. <i>Hardware</i>	59
4.1.4. <i>Software</i>	59
4.2. Análisis de resultados	60
4.2.1. Obtención de resultados	61
4.2.2. Métricas	62
4.2.3. Análisis de resultados	64
5. Conclusiones y trabajo futuro	73
5.1. Conclusiones	73
5.2. Conclusions	75
5.3. Trabajo Futuro	76
5.3.1. Color Quantization using K-Means	76
5.3.2. Detección en vídeo	79
5.3.3. Mejora de la detección en condiciones adversas	80
5.3.4. Inclusión de sistemas de geolocalización	80
5.3.5. Automatización y sistema de alertas	80
5.3.6. App orientada a cuerpos de emergencia	81

Contribuciones Personales	83
Apéndice: Manual de Usuario	91

Índice de figuras

2.1.	Representación de tensores de diferentes dimensiones	16
2.2.	Representación gráfica del ejemplo	17
2.3.	Puntos críticos	18
2.4.	Ejemplo de convolución 2D	23
2.5.	Convolución con padding y stride	24
2.6.	Pooling con elementos fuera de la ventana	25
2.7.	Pooling sin elementos fuera de ventana gracias a padding	25
2.8.	Diferentes comportamientos en función del ajuste	26
2.9.	Detección temprana en error contra iteraciones	27
2.10.	Arquitectura general de la detección de objetos	29
2.11.	Intersección sobre la Unión formula	29
2.12.	$s_0 = 1$	30
2.13.	$s_1 = 2$	30
2.14.	$s_2 = 0,5$	30
2.15.	Indices <i>Ground truth</i>	31
2.16.	<i>Bounding Boxes</i> solapadas	32
2.17.	<i>Bounding Boxes</i> finales	32
2.18.	Capas red Yolo	34
2.19.	Arquitectura red <i>YoloX</i> .[1]	37
3.1.	Diagrama de casos de uso	40
3.2.	Ventana inicial de la aplicación	41
3.3.	Menú principal con las opciones de entrenamiento y clasificación	42
3.4.	Selección de parámetros de entrenamiento	43
3.5.	Selección de modelo para clasificar	46
3.6.	Selección de parámetros del modelo	46
3.7.	Selección de opciones para la clasificación	47
3.8.	Resultado final	47
4.1.	Ejemplo de entorno nevado	57
4.2.	Ejemplo de pradera	57
4.3.	Ejemplo de bosque frondoso	58
4.4.	Ejemplo de llanura	58
4.5.	Visualización división del dataset, Fuente: Propia	61

4.6.	Proceso de entrenamiento y validación, Fuente: Propia	62
4.7.	Ejemplo de la Curva <i>Precision Recall</i>	63
4.8.	Ejemplo de evaluación de métricas en entrenamiento de un modelo YOLOv4	65
4.9.	Ejemplo de evaluación de métricas en entrenamiento de un modelo YOLOv4 de 100 épocas	66
4.10.	Resultado obtenido por el mejor modelo YOLOv4 para el conjunto de test .	67
4.11.	Ejemplo de evaluación de métricas en el proceso de entrenamiento de un modelo <i>YOLOX</i>	68
4.12.	Resultado obtenido por el mejor modelo <i>YOLOX</i> para el conjunto de test .	69
4.13.	Comparación de la imagen 667 etiquetada por <i>YOLOv4</i> y <i>YOLOX</i>	70
4.14.	Comparación de la imagen 1259 etiquetada por <i>YOLOv4</i> y <i>YOLOX</i>	71
4.15.	Comparación de la imagen 824 etiquetada por <i>YOLOv4</i> y <i>YOLOX</i>	71
4.16.	Comparación de la imagen 1259 en condiciones de nieve, etiquetada por <i>YOLOv4</i> y <i>YOLOX</i>	72
5.1.	Ejemplo de aplicación de algoritmo de clustering, Fuente: Propia	77
5.2.	Imagen original	78
5.3.	Imagen tras aplicarle K-Means	78
4.	Panel Matlab	92
5.	Botón Run	92

Capítulo 1

Introducción

El presente capítulo tiene como objetivo proporcionar el contexto necesario para comprender la problemática abordada en este trabajo de fin de grado. Se presentan los antecedentes que justifican la relevancia del estudio, destacando la importancia de la localización eficiente de personas en situaciones de emergencia y el papel que las nuevas tecnologías pueden desempeñar en este ámbito.

Además, se describe la motivación que ha impulsado la realización de este proyecto, tanto desde un punto de vista técnico como humanitario. También se establecen los objetivos generales y específicos que se persiguen con el desarrollo de este trabajo, así como la planificación seguida para su implementación. Finalmente, se discuten las limitaciones y el alcance del proyecto, proporcionando una visión clara de sus posibilidades y posibles mejoras futuras.

En los siguientes apartados, se detallarán cada uno de estos aspectos, proporcionando un marco de referencia sólido para el desarrollo del resto de la memoria.

1.1. Antecedentes

Las operaciones de búsqueda y rescate (SAR) en entornos montañosos o remotos enfrentan desafíos únicos: terrenos abruptos, condiciones meteorológicas cambiantes y la dificultad de cubrir grandes áreas en poco tiempo. En España, regiones como los Pirineos o la cordillera Cantábrica concentran un alto número de emergencias donde estos factores retrasan la localización de víctimas, aumentando el riesgo de desenlaces fatales. Según datos del GREIM[2] (Grupos de Rescate Especial de Intervención en Montaña) de la Guardia Civil, durante los primeros siete meses de 2023 se registraron un total de 633 operaciones de rescate. De estas, el 58.14 % fueron causadas por accidentes provocados por la sobreestimación de las capacidades físicas o técnicas de los excursionistas. Un dato especialmente relevante es que en el 20.85 % de los casos (132 accidentes) fue necesario realizar labores de búsqueda, lo que significa que en uno de cada cinco accidentes, la localización exacta de los afectados era desconocida. Esto subraya la importancia de contar con herramientas

y métodos eficaces para agilizar las tareas de localización y rescate.

Los grupos de rescate, como el GREIM, disponen de recursos tales como efectivos especializados, vehículos todoterreno y perros de búsqueda, que son fundamentales para facilitar la localización de personas en situaciones de emergencia. Sin embargo, a pesar de estos recursos, el tiempo sigue siendo un factor crítico en las operaciones de rescate. En muchas ocasiones, la diferencia entre el éxito y el fracaso de una misión de rescate depende de la rapidez con la que se pueda detectar a los afectados, especialmente en entornos hostiles o de difícil acceso.

A lo largo de los años, las técnicas y herramientas utilizadas en las operaciones de rescate han evolucionado significativamente. Inicialmente, los rescates dependían casi exclusivamente de la experiencia y el conocimiento del terreno por parte de los equipos de rescate, así como del uso de perros entrenados para rastrear personas. Con el tiempo, se incorporaron tecnologías como los sistemas de comunicación por radio, que permitieron una mejor coordinación entre los equipos, y el uso de helicópteros, que facilitaron el acceso a zonas remotas y la visualización de áreas extensas desde el aire.

No obstante, estas metodologías tradicionales presentan limitaciones, especialmente en terrenos con orografía compleja o en áreas con vegetación densa. En estos casos, el acceso a pie puede ser extremadamente lento y peligroso, mientras que los medios aéreos tradicionales como helicópteros pueden tener dificultades para obtener una visión clara del terreno debido a la altura o a la presencia de obstáculos naturales. Es aquí donde las nuevas tecnologías, como los drones equipados con sistemas de detección basados en inteligencia artificial, pueden marcar la diferencia.

Los vehículos aéreos no tripulados (UAVs o drones) han surgido como una solución prometedora para superar estas barreras. Su capacidad para operar a bajas altitudes, acceder a áreas inaccesibles y combinar múltiples sensores (cámaras RGB, térmicas, LiDAR) los convierte en herramientas ideales para SAR. Estudios recientes respaldan su potencial;

Por ejemplo, Nguyen et al. (2020)[3] demostraron que modelos de aprendizaje profundo como YOLOv3 pueden detectar personas en imágenes térmicas capturadas por drones con una precisión superior al 90 %, incluso en condiciones de poca visibilidad. Otro trabajo relevante, publicado por Sampedro et al.[4], revisa diferentes técnicas de visión por computador aplicadas a drones, destacando su eficacia en entornos montañosos y boscosos.

Un caso de éxito destacable es el proyecto LARICS (*Laboratory for Robotics and Intelligent Control Systems*)[5], desarrollado por la Universidad de Zagreb (Croacia), se centró en mejorar la eficiencia de las operaciones de búsqueda y rescate mediante el uso de drones equipados con sistemas de inteligencia artificial y sensores térmicos, más concretamente con YOLOv4. Las pruebas se realizaron en los Alpes Dináricos, un terreno con bosques densos, con desniveles abruptos y condiciones meteorológicas variables. El sistema alcanzó un 92 % de precisión en la detección de humanos, incluso con obstrucciones parciales. En España, el GREIM ya ha implementado pruebas piloto con drones en rescates reales, como en el Parque Nacional de Ordesa[6] en 2023.

Pese a sus ventajas, los drones enfrentan diversos desafíos, a saber: autonomía limitada

de batería (30-50 minutos en la mayoría de modelos), interferencias en bosques densos y la necesidad de operadores certificados. Además, la normativa europea EASA[7] restringe vuelos en espacios aéreos controlados.

1.2. Motivación

Durante la realización de los grados, los autores del presente trabajo han presentado cierta curiosidad y afinidad hacia asignaturas relacionadas con la Inteligencia Artificial. Dichas asignaturas se centran principalmente en trabajar con datos tabulares y en algunos casos con texto si se trataba de trabajar con procesamiento de lenguaje natural. Dichas asignaturas han permitido a los autores adquirir conocimientos acerca del entrenamiento de modelos de aprendizaje automático, y el posterior análisis de resultados de los mismos. La mayoría de trabajos y prácticas estaban centrados en aplicar métodos de aprendizaje automático a datos numéricos, sin embargo no se ha profundizado en aplicar los métodos de aprendizaje automático en contenido multimedia como pueden ser las imágenes. Es por eso que este proyecto generó interés desde un punto de vista técnico.

En segundo lugar, dejando de lado la componente puramente científica o académica cabe destacar la motivación humanitaria que este tipo de proyectos presentan. Gracias a los avances en el desarrollo de técnicas de aprendizaje profundo, especialmente con imágenes, se pueden construir sistemas realmente sofisticados que sean capaces de cooperar en situaciones críticas donde cualquier ayuda es bienvenida y donde pequeños detalles pueden marcar la diferencia. En el caso de este proyecto son evidentes las potenciales contribuciones del sistema. Hablamos de modelos capaces de reconocer siluetas de seres humanos en entornos naturales mediante fotografías tomadas por drones a alturas de alrededor de 30 metros del suelo.

Este tipo de tecnología puede ser de gran utilidad en operaciones de rescate y búsqueda, donde la detección temprana de personas en cualquier entorno, sobre todo en entornos complejos o de difícil acceso, puede salvar vidas. Algunos ejemplos son en accidentes en zonas montañosas, situaciones de desastres naturales o áreas de difícil acceso. En todas estas situaciones la capacidad de detectar rápidamente la ubicación de personas mediante imágenes aéreas puede agilizar y facilitar en gran medida las labores de rescate reduciendo el tiempo de respuesta y aumentando las probabilidades de supervivencia de las personas afectadas.

Además, el uso de drones equipados con sistemas de detección visual ofrece ventajas significativas frente a otros métodos más tradicionales, como la búsqueda manual o el uso de helicópteros. Los drones pueden cubrir grandes áreas en menos tiempo, acceder a terrenos más complicados o más pequeños en los que un helicóptero no podría acceder, pueden proporcionar imágenes detalladas en tiempo real y económicamente son accesibles por cualquier organización, lo cual no ocurre con otros medios aéreos.

1.3. Objetivos

Los objetivos principales del proyecto se centran en descubrir, comprender y aplicar métodos de aprendizaje profundo aplicados en la detección de elementos en imágenes, específicamente en la detección de seres humanos. Para ello se pretende investigar y familiarizarse con métodos como *YOLO* (*You Only Look Once*), en sus diferentes versiones, lo cual permitirá evaluar el comportamiento de las mismas, tanto entre ellas como en función de la elección de hiperparámetros. Estos hiperparámetros, que pueden modificar el comportamiento y los resultados de los modelos, ofrecen una amplia gama de posibilidades para mejorar el rendimiento. Esto también permitirá obtener conclusiones y dotar a los usuarios con las herramientas necesarias para abordar el problema de manera efectiva y obtener los mejores resultados posibles.

Un segundo objetivo del proyecto es la realización de una aplicación que permita a los usuarios interactuar con varios modelos de detección de objetos, así como entrenar nuevos. Esto dotará de experiencia y conocimientos a los autores de todo aquello que debe tenerse en cuenta a la hora de crear una aplicación cuyo nicho es el aprendizaje profundo, tipos de aplicaciones que requieren de requisitos y funcionalidades diferentes a otras realizadas durante los estudios.

Por último, debido al volumen de trabajo y la complejidad del proyecto, se busca la aplicación de técnicas de organización, ingeniería de requisitos y desarrollo de software adquiridas durante su formación académica. Esto incluye la planificación de tareas, la gestión de recursos, la documentación del proceso y la implementación de buena práctica en el desarrollo de software.

En resumen, los objetivos específicos se concretan en lo siguiente:

1. Adquirir conocimientos mediante la comprensión de la tecnología de la Inteligencia Artificial y su aplicación mediante el Aprendizaje Profundo.
2. Conocer y aprender el funcionamiento de modelos de detección de objetos, basados en CNN, explorando su aplicación en el reconocimiento de personas en imágenes capturadas desde un dron.
3. Desarrollar una aplicación intuitiva para usuarios con diferentes niveles de experiencia. Desde aquellos con poca familiaridad en las tecnologías, hasta quienes posean nociones más avanzadas.
4. Integrar todas las tecnologías funcionales bajo una interfaz amigable, realizando tanto pruebas individuales como de integración.
5. Incluir un manual de usuario que permita entender el funcionamiento de la aplicación y permita a los usuarios con menos nociones sobre las tecnologías usadas, aprender un concepto introductorio de ellas y su funcionamiento.
6. Experimentar con diferentes modelos de detección de objetos observando sus resultados y comparándolos para poder entender el funcionamiento de cada modelo.

7. Generar una documentación detallada del trabajo, que recoja la arquitectura de la aplicación, el proceso de desarrollo, los modelos utilizados y los resultados obtenidos. Facilitando la comprensión del proyecto.

1.4. Plan de trabajo

La realización de un plan de trabajo ha sido fundamental para garantizar la correcta ejecución del proyecto. Nos ha permitido organizar el trabajo o tareas a realizar para garantizar el cumplimiento de los objetivos según el plazo establecido. A continuación, se muestran las diversas fases realizadas para completar el proyecto.

1. **Elección del tema:** Aunque esta fase se realizó anteriormente a la inicialización del proyecto, es importante recalcarla, ya que marca el inicio de este proyecto. Durante esta fase se propusieron diversos temas y tras un pequeño estudio de consideraciones técnicas que cada proyecto pudiese aportar a los estudiantes y motivación de la temática al final se eligió el actual.
2. **Elección del entorno de desarrollo:** Se realizó un estudio de dos posibles lenguajes en los cuales se podían llevar a cabo el desarrollo. Por un lado, *Python* y por otro *Matlab*. Ambos permiten la implementación de reconocimiento de imágenes utilizando modelos CNN. *Python*, mediante el uso de *TensorFlow*, era una opción atractiva sobre todo porque algunos de los integrantes del equipo ya contaban con conocimientos de dicho *framework*¹ de redes neuronales implementadas en *Python*. Además de presentar una sencilla integración con otras tecnologías de *frontend*² para desarrollar la aplicación.

Matlab aunque fuese una experiencia totalmente nueva para el equipo de desarrollo, también resultaba una opción interesante. Cuenta con varios módulos que permiten la implementación de redes neuronales, módulos como *Deep Learning Toolbox* entre otros. También nos ofrecía la opción de diseñar la interfaz gráfica mediante *Matlab App Designer*.

Al final, tras evaluar las ventajas e inconvenientes de ambas opciones, se eligió *Matlab*, ya que tiene todo lo necesario implementado y no tendríamos que depender de diversas librerías que podían ser modificadas con el tiempo, como ocurre en *Python*, o tener que integrar el código con otras herramientas para realizar la interfaz gráfica.

Además, ofrece documentación detallada sobre las funcionalidades que abarca y las fuentes de referencia en las que se fundamenta. Por otro lado, la Universidad Complutense cuenta con una licencia para su uso, lo que garantiza el acceso al soporte técnico necesario en caso de ser requerido.

¹Un *framework* es un conjunto de herramientas y componentes reutilizables que proporcionan una estructura de trabajo para facilitar el desarrollo y mantenimiento de software.

²El *frontend* es la parte de una aplicación o sitio web con la que interactúa directamente el usuario, e incluye el diseño, la estructura visual y la experiencia de usuario.

3. **Estudio y selección de las Redes Neuronales Convolucionales:** Tras haber elegido el entorno de desarrollo a utilizar se tuvo que elegir el tipo de redes neuronales a utilizar. Existen una gran cantidad de modelos de redes neuronales como por ejemplo *AlexNet*, *GoogLeNet* centradas en la clasificación de imágenes, pero las características del proyecto requerían centrarse en la detección de objetos, es por eso que se decidió explorar la familia de detectores *YOLO*, en concreto *YOLOv4* y *YOLOX*.

Como se verá más adelante, este detector pertenece a la categoría de los denominados *single-stage detectors*, ampliamente utilizados en aplicaciones en tiempo real. Una de sus principales ventajas es que requiere pocos recursos computacionales, lo que lo hace ideal para dispositivos con capacidad limitada, como los sistemas a bordo de drones.

4. **Búsqueda de datasets³:** Durante esta fase se llevó a cabo la búsqueda de conjuntos de datos que cumplieran con una serie de requisitos específicos: que estuvieran orientados a la detección de personas, que las imágenes hubieran sido tomadas desde un dron, que contaran con una licencia de uso adecuada, y que incluyeran anotaciones precisas, es decir, que indicaran la localización de las personas en cada imagen. Estos criterios limitaron considerablemente las opciones disponibles, lo que dificultó el proceso de búsqueda. No obstante, se logró identificar un dataset que satisfacía todas las condiciones establecidas y que, además, disponía de una gran cantidad de imágenes, lo cual resultaba especialmente beneficioso para proporcionar mayor variedad durante el entrenamiento de la red neuronal.
5. **Desarrollo del código de la aplicación:** En esta fase se procedió a un estudio más detallado de los modelos que se iban a emplear, así como a la búsqueda de ejemplos que sirvieran de referencia. Para poder utilizar las imágenes del conjunto de datos con *YOLO*, fue necesario redimensionarlas a un formato compatible. Asimismo, se realizó una conversión de las anotaciones de los *bounding boxes*⁴ para adaptarlas al formato requerido por el modelo. Una vez preparado el modelo, las imágenes se dividieron en conjuntos de entrenamiento y validación. Finalmente, se inició el entrenamiento de distintos modelos utilizando diferentes configuraciones de parámetros.
6. **Desarrollo de la interfaz gráfica:** Para realizar la interfaz gráfica se eligió utilizar la herramienta que ofrece *Matlab*, *Matlab App Designer* la cual está centrada en el diseño de interfaces y es bastante simple y fácil de utilizar. Tiene un panel en la parte izquierda que permite seleccionar diversos elementos como botones, cuadros de texto o imágenes entre otros. También cuenta con un panel central, el cual representa la propia aplicación, donde al ir colocando los diferentes elementos se consigue ir creando la aplicación, con todas las pantallas y elementos necesarios. A su vez, también cuenta con un apartado de código, donde se pueden crear funcionalidades a los diferentes elementos, para que al pulsar cierto botón pueda realizar una acción, lo que permite implementar dinamismo dentro de la propia aplicación y no tener que depender de otras herramientas.
7. **Estudio de los resultados obtenidos:** Una vez terminadas las fases anteriores se planifica la comparación de varios de los modelos entrenados para determinar

³Un *dataset* es un conjunto de datos estructurados que se utiliza para entrenar, validar o probar modelos en tareas de aprendizaje automático o análisis de datos.

⁴Las *bounding boxes* son rectángulos delimitadores que se utilizan para indicar la ubicación de un objeto dentro de una imagen, generalmente mediante sus coordenadas y dimensiones.

cuáles eran los más precisos o los que mejores resultados generaban. Tomados estos modelos se determina la realización de pequeños ajustes en sus parámetros para intentar conseguir un modelo más preciso.

8. **Finalización de la interfaz:** Una vez definidos todos los modelos que formaban parte de la aplicación, se completa el diseño de la misma, incorporando los apartados y opciones necesarios para llevar a cabo la detección utilizando los modelos previamente entrenados. Asimismo, se finaliza el diseño del módulo destinado al entrenamiento de un modelo directamente desde la propia aplicación.

1.5. Alcance y limitaciones

El alcance de la aplicación se encuentra limitado por el tipo de imágenes que pueden utilizarse. Estas deben ser similares a las empleadas durante el entrenamiento de los modelos, ya que dicho entrenamiento se realizó con imágenes capturadas por el dron, en las que las personas aparecen vistas desde una perspectiva cenital. Como consecuencia, al intentar detectar personas desde otros ángulos, es probable que el sistema no ofrezca resultados precisos. Sin embargo, es importante señalar que este alcance podría ampliarse incorporando una mayor variedad de imágenes tomadas desde distintas perspectivas. Esta diversidad permitiría entrenar modelos más robustos y versátiles, capaces de detectar personas en entornos y ángulos más variados.

Para compensar este déficit de imágenes, se puede aplicar la técnica conocida como *Image Augmentation*, que permite generar nuevas imágenes a partir de las existentes sin necesidad de recolectar datos adicionales. Esta técnica consiste en realizar diversas transformaciones sobre las imágenes originales, como rotaciones, escalado, cambios en la iluminación, entre otras modificaciones, con el objetivo de aumentar la diversidad del conjunto de datos y mejorar la capacidad de generalización del modelo.

1.6. Organización de la Memoria

Habiendo expuesto la motivación para el desarrollo de la aplicación, definidos los objetivos y el plan de trabajo, el resto de la memoria se organiza de la siguiente manera.

En el **Capítulo 2** se abordan los conceptos, métodos y técnicas que sustentan el desarrollo del proyecto, centrándose en las redes neuronales convolucionales y, en particular, en los detectores de objetos de la familia *YOLO*, como *YOLOv4* y *YOLOX*.

El **Capítulo 3** está dedicado al proceso de diseño y desarrollo de la aplicación. Se describen aspectos como la arquitectura, los módulos desarrollados, los casos de uso y la interfaz de usuario, así como los planteamientos metodológicos que han guiado la implementación.

En el **Capítulo 4** se detalla el conjunto de datos, es decir, el *dataset* utilizado para el

entrenamiento y validación de los modelos, así como los resultados obtenidos a partir de diferentes configuraciones. Se analizan las métricas empleadas y se comparan los distintos modelos con el fin de evaluar su rendimiento.

Finalmente, en el **Capítulo 5** se presentan las conclusiones extraídas del trabajo realizado y se plantea una propuesta de líneas futuras de mejora y extensión del proyecto, tanto a nivel técnico como funcional.

1.7. Introduction

The aim of this chapter is to provide the necessary context to understand the problem addressed in this thesis. It presents the background that justifies the relevance of the study, highlighting the importance of the efficient location of people in emergency situations and the role that new technologies can play in this field.

In addition, the motivation behind the project is described, both from a technical and humanitarian point of view. It also sets out the general and specific objectives pursued in the development of this work, as well as the planning followed for its implementation. Finally, the limitations and scope of the project are discussed, providing a clear vision of its possibilities and possible future improvements.

In the following sections, each of these aspects will be detailed, providing a solid frame of reference for the development of the rest of the report.

1.8. Background

Search and Rescue (SAR) operations in mountainous or remote environments face unique challenges: rugged terrain, rapidly changing weather conditions, and the difficulty of covering large areas in a short time. In Spain, regions such as the Pyrenees or the Cantabrian mountain range account for a high number of emergencies where these factors delay the location of victims, increasing the risk of fatal outcomes. According to data from the GREIM[2] (Special Mountain Intervention Rescue Groups) of the Civil Guard, a total of 633 rescue operations were recorded during the first seven months of 2023. Of these, 58.14 % were caused by accidents resulting from overestimation of the hikers' physical or technical abilities. A particularly relevant statistic is that in 20.85 % of cases (132 accidents), search efforts were necessary, meaning that in one out of every five incidents, the exact location of those affected was unknown. This highlights the importance of having effective tools and methods to expedite search and rescue tasks.

Rescue groups, such as GREIM, are equipped with specialized personnel, all-terrain vehicles, and search dogs, all of which are essential to facilitate the location of people in emergency situations. However, despite these resources, time remains a critical factor in rescue operations. In many cases, the difference between the success and failure of a

rescue mission depends on how quickly the victims can be located, especially in hostile or difficult-to-access environments.

Over the years, the techniques and tools used in rescue operations have evolved significantly. Initially, rescues depended almost exclusively on the experience and terrain knowledge of the rescue teams, as well as the use of trained dogs to track individuals. Over time, technologies such as radio communication systems were introduced, improving coordination between teams, and helicopters were used to access remote areas and observe large regions from above.

Nevertheless, these traditional methodologies have limitations, particularly in terrains with complex orography or areas with dense vegetation. In such cases, access on foot can be extremely slow and dangerous, while traditional aerial means like helicopters may struggle to obtain a clear view of the terrain due to altitude or the presence of natural obstacles. This is where new technologies, such as drones equipped with artificial intelligence-based detection systems, can make a significant difference.

Unmanned Aerial Vehicles (UAVs or drones) have emerged as a promising solution to overcome these barriers. Their ability to operate at low altitudes, access inaccessible areas, and combine multiple sensors (RGB cameras, thermal imaging, LiDAR) makes them ideal tools for SAR missions. Recent studies support their potential:

For example, Nguyen et al. (2020)[3] demonstrated that deep learning models such as YOLOv3 can detect people in thermal images captured by drones with over 90 % accuracy, even under low-visibility conditions. Another notable study, published in MDPI Remote Sensing by Sampedro et al. (2019)[4], reviews various computer vision techniques applied to drones, highlighting their effectiveness in mountainous and forested environments.

A notable success case is the LARICS project (Laboratory for Robotics and Intelligent Control Systems)[5], developed by the University of Zagreb (Croatia), which focused on improving the efficiency of search and rescue operations through the use of drones equipped with artificial intelligence systems and thermal sensors, specifically YOLOv4. The tests were conducted in the Dinaric Alps, a region characterized by dense forests, steep slopes, and variable weather conditions. The system achieved 92 % accuracy in human detection, even with partial obstructions. In Spain, GREIM has already implemented pilot tests with drones in real rescue operations, such as in Ordesa National Park[6] in 2023.

Despite their advantages, drones face challenges: limited battery life (30–50 minutes in most models), interference in dense forests, and the need for certified operators. Additionally, European Union Aviation Safety Agency (EASA) regulations[7] restrict flights in controlled airspace.

1.9. Motivation

During their studies, the authors of this thesis have shown a certain curiosity and affinity towards subjects related to Artificial Intelligence. These subjects are mainly focused

on working with tabular data and, in some cases, with text when working with natural language processing. These subjects have allowed the students to acquire knowledge about the training of machine learning models, and the subsequent analysis of their results. Most of the work and practices were focused on applying machine learning methods to numerical data, but there has been no in-depth study of the application of machine learning methods to multimedia content such as images. That is why this project generated interest from a technical point of view.

Secondly, leaving aside the purely scientific or academic component, it is worth highlighting the humanitarian motivation of this type of project. Thanks to advances in the development of deep learning techniques, especially with images, it is possible to build truly sophisticated systems that are capable of cooperating in critical situations where any help is welcome and where small details can make a difference. In the case of this project, the potential contributions of the system are evident. We are talking about models capable of recognising silhouettes of human beings in natural environments using photographs taken by drones at heights of around 30 metres above the ground.

This type of technology can be of great use in rescue and search operations, where early detection of people in any environment, especially in complex or difficult to access environments, can save lives. Examples include accidents in mountainous areas, natural disaster situations or areas that are difficult to access. In all these situations, the ability to quickly detect the location of people using aerial imagery can greatly speed up and facilitate rescue efforts by reducing response time and increasing the chances of survival of those affected.

In addition, the use of drones equipped with visual detection systems offers significant advantages over more traditional methods such as manual search or the use of helicopters. Drones can cover large areas in less time, can access smaller or more complicated terrain that a helicopter could not access, can provide detailed images in real time and are cost-effectively accessible by any organisation, which is not the case with other aerial means.

1.10. Objectives

The main objectives of the project focus on discovering, understanding and applying deep learning methods applied to the detection of elements in images, specifically in the detection of human beings. To this end, the aim is to investigate and become familiar with methods such as YOLO (You Only Look Once), in its different versions, which will make it possible to evaluate the behaviour of these methods, both between them and in terms of the choice of hyperparameters. These hyperparameters, which can modify the behaviour and results of the models, offer a wide range of possibilities for improving performance. This will also allow conclusions to be drawn and provide users with the necessary tools to tackle the problem effectively and obtain the best possible results.

A second objective of the project is the realisation of an application that allows users to interact with various object detection models, as well as to train new ones. This will provide the students with experience and knowledge of what needs to be taken into account when

creating a niche application for deep learning, types of applications that have different requirements and functionalities than others developed during the studies.

Finally, due to the volume of work and complexity of the project, the application of organisational, requirements engineering and software development techniques acquired during their academic training is sought. This includes task planning, resource management, process documentation and the implementation of good practice in software development.

In summary, the specific objectives are as follows:

1. Acquire knowledge through the understanding of Artificial Intelligence technology and its application via Deep Learning.
2. Learn and understand how object detection models based on CNNs work, exploring their application in the recognition of people in images captured by drones.
3. Develop an intuitive application for users with varying levels of experience, from those with little familiarity with the technologies to those with more advanced knowledge.
4. Integrate all functional technologies under a user-friendly interface, conducting both individual and integration testing.
5. Include a user manual that explains the operation of the application and helps users with limited technical background understand the introductory concepts of the technologies used.
6. Experiment with different object detection models, analyzing and comparing their results to understand how each model works.
7. Produce detailed documentation of the work, covering the application's architecture, the development process, the models used, and the results obtained, to facilitate understanding of the project.

1.11. Work Plan

The development of a work plan has been essential to ensure the proper execution of the project. It allowed us to organize the tasks to be carried out and ensure that the objectives were met within the established timeframe. Below are the various phases completed to carry out the project:

1. **Topic selection:** Although this phase took place prior to the official start of the project, it is important to highlight it, as it marks the beginning of the entire process. During this stage, various topics were proposed. After a brief analysis of the technical aspects each project could offer to the students, as well as the motivation for the chosen subject, the current topic was selected.

- 2. Development environment selection:** A comparison was made between two possible programming languages in which the development could be carried out: *Python* and *Matlab*. Both allow the implementation of image recognition using CNN models.

Python, with the use of *TensorFlow*, was an attractive option, particularly because some team members were already familiar with this *framework*⁵ for neural networks implemented in *Python*. Additionally, it offers easy integration with other *frontend* technologies⁶ for building the application interface.

Matlab, although a completely new experience for the development team, also proved to be an interesting choice. It includes several modules that support the implementation of neural networks, such as the *Deep Learning Toolbox*, among others. It also provides the option to design the graphical user interface using *Matlab App Designer*.

In the end, after evaluating the pros and cons of both options, *Matlab* was chosen because it offers everything needed within a single environment, without relying on multiple external libraries that may change over time (as in the case of Python), or requiring the integration of code with other tools to create the user interface.

Additionally, it provides detailed documentation on its functionalities and the reference sources on which it is based. Furthermore, the Complutense University holds a license for its use, ensuring access to the necessary technical support if needed.

- 3. Study and selection of Convolutional Neural Networks (CNNs):** After selecting the development environment, the next step was to choose the type of neural networks to use. There are many types of neural network models, such as *AlexNet* and *GoogLeNet*, which focus on image classification. However, the nature of this project required a focus on object detection, which led to the exploration of the YOLO family of detectors, specifically Yolov4 and YoloX.

This detector belongs to the category known as single-stage detectors, which are widely used in real-time applications. One of their main advantages is their low computational resource requirement, making them ideal for use on low-capacity devices, such as drone onboard systems.

- 4. Search for datasets⁷:** In this phase, a search was conducted for datasets that met a set of specific criteria: they had to be focused on human detection, contain images captured from drones, include an appropriate usage license, and provide precise annotations—that is, indicate the location of individuals in each image. These requirements significantly narrowed down the available options, making the search process more challenging. Nevertheless, a dataset was found that met all the specified conditions and contained a large number of images, which was particularly beneficial for providing variety during neural network training.

- 5. Application code development:** This phase involved a more in-depth study of the models to be used, as well as the search for example implementations to use as references. In order to use the dataset images with YOLO, the images had to be resized to a compatible format. Additionally, the annotations of the *bounding*

⁵A *framework* is a set of reusable tools and components that provides a structured environment to facilitate software development and maintenance.

⁶The *frontend* is the part of an application or website with which users directly interact, and includes the visual layout, design, and user experience.

⁷A *dataset* is a structured collection of data used for training, validating, or testing models in machine learning or data analysis tasks.

*boxes*⁸ had to be converted to the format required by the model. Once the model was prepared, the images were divided into training and validation sets. Finally, training was initiated on several models using different parameter configurations.

6. **Graphical user interface development:** For the graphical interface, the *Matlab App Designer* tool was chosen. This tool is specifically designed for interface development and is relatively simple and easy to use. It includes a panel on the left side that allows the selection of various elements such as buttons, text boxes, and images, among others. The central panel represents the actual application, where elements are placed to construct the interface, including all necessary screens and components. Additionally, there is a coding section where functionalities for each element can be created, enabling actions such as executing a function when a button is pressed. This adds interactivity to the application and eliminates the need to rely on external tools.
7. **Analysis of obtained results:** Once the previous phases were completed, the next step was to compare several of the trained models to determine which produced the most accurate or effective results. Based on these results, minor adjustments were made to the model parameters to attempt to improve their precision.
8. **Finalization of the interface:** After defining all the models included in the application, the interface was finalized by adding all necessary sections and options to perform detections using the previously trained models. Additionally, the design of the module for training a model directly within the application was completed.

1.12. Scope and Limitations

The scope of the application is limited by the type of images that can be used. These must be similar to those used during the training of the models, as the training was carried out with images captured by the drone, in which people are seen from a zenithal perspective. As a consequence, when trying to detect people from other angles, the system is likely to give inaccurate results. However, it is important to note that this range could be extended by incorporating a wider variety of images taken from different perspectives. This diversity would allow more robust and versatile models to be trained, capable of detecting people in more varied environments and angles.

To compensate for this lack of images, the technique known as Image Augmentation can be applied, which allows new images to be generated from existing ones without the need to collect additional data. This technique consists of performing various transformations on the original images, such as rotations, scaling, changes in illumination, among other modifications, with the aim of increasing the diversity of the data set and improving the generalisation capacity of the model.

⁸*Bounding boxes* are rectangular delimiters used to indicate the location of an object within an image, typically using coordinates and dimensions.

1.13. Thesis Organization

Having outlined the motivation behind the development of the application, defined the objectives, and established the work plan, the remainder of this report is organized as follows:

Chapter 2 addresses the concepts, methods, and techniques that support the development of the project, focusing on convolutional neural networks and, in particular, on object detectors from the YOLO family, such as *YOLOv4* and *YOLOX*.

Chapter 3 is dedicated to the design and development process of the application. It describes aspects such as the architecture, developed modules, use cases, and the user interface, as well as the methodological approaches that guided the implementation.

Chapter 4 provides details about the dataset used for training and validating the models, along with the results obtained from different configurations. It analyzes the metrics used and compares the various models to assess their performance.

Finally, **Chapter 5** presents the conclusions drawn from the work carried out and outlines proposals for future improvements and extensions of the project, both technically and functionally.

Capítulo 2

Conceptos, métodos y técnicas aplicadas

2.1. Introducción

En este capítulo se explican con detenimiento los métodos, técnicas y operaciones necesarias para llevar a cabo la realización del proyecto. Estos conceptos abordan temáticas como la computación numérica y el aprendizaje automático, centrado en modelos CNN, especialmente en *YOLO*, para la detección de objetos en imágenes. Se explican sus fundamentos teóricos/matemáticos y se profundiza en aquellos que han sido seleccionados para el desarrollo del trabajo aquí expuesto. La base del conocimiento empleado para la realización de este capítulo se ha tomado del libro Pajares et al. (2021)

2.2. Tensores

Los tensores son una de las estructuras de datos, Figura 2.1, de mayor relevancia a la hora de diseñar modelos de aprendizaje automático. Un tensor es un objeto algebraico relacionado con un espacio vectorial. Los vectores y los escalares (comúnmente utilizados en física elemental) son los tensores más simples.

Desde un punto de vista formal, un tensor T es una matriz generalizada con una serie de números dispuestos en una cuadrícula regular con un número variable de ejes. Podemos definir un tensor de tres dimensiones como $T_{i,j,k}$ o de cuatro como $T_{i,j,k,l}$. En la Figura 2.1, se muestra la representación de un tensor de una, dos, tres y cuatro dimensiones respectivamente.

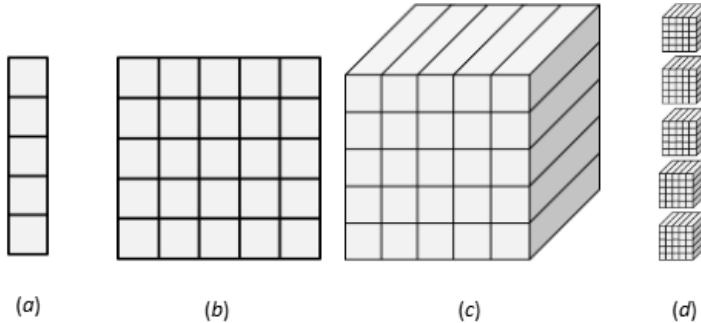


Figura 2.1: Representación de tensores de diferentes dimensiones.

2.3. Métodos de optimización

2.3.1. Concepto de optimización

El concepto de optimización es trascendental a la hora de comprender el comportamiento de los métodos destinados a la predicción. La optimización se refiere a la tarea de minimizar o maximizar una función $f(x)$ modificando x . Por lo general, la optimización se asocia con la minimización, ya que la maximización es equivalente a la minimización de $-f(x)$.

La función a minimizar se denomina *función objetivo*, y se emplea para evaluar una solución propuesta (candidata), como ocurre con los pesos de las redes neuronales. Cuando se está minimizando, según Goodfellow y col.[8], se le denomina también *función de coste* (*cost function*), *función de pérdida* (*loss function*) o también *función de error* (*error function*). Uno de los beneficios que otorga la función de pérdida es la de obtener un error que poder retropropagar en la etapa de aprendizaje. De esta forma, se pueden actualizar los valores necesarios para que dicha función alcance el menor valor posible. El valor que obtiene la minimización o maximización se identifica con el símbolo $*$, siendo:

$$x^* = \arg \min f(x) \quad (2.1)$$

Cabe destacar que dicha función debe ser derivable y que, durante el proceso de entrenamiento, se compara la salida del modelo (predicción) con la etiqueta verdadera (*ground-truth*).

Resulta bien conocido que la derivada de una función en un punto x proporciona la pendiente de la función en dicho punto. La derivada permite generar pequeños cambios en la entrada (x) de forma que se pueda obtener el correspondiente cambio en la salida (y). Se sabe que $f(x - \varepsilon \operatorname{sign}(f'(x)))$ es menor que $f(x)$ para un ε lo suficientemente pequeño. Se puede entonces disminuir $f(x)$ moviendo x en pequeños pasos con el signo opuesto de la derivada. Esto se conoce como gradiente descendente (gradient descent). Veamos un ejemplo:

Tomamos como función $f(x) = \frac{1}{2}x^2$ que posee como derivada $f'(x) = x$ de forma que se pueden tener las siguientes consideraciones:

- Para $x < 0$, $f'(x) < 0$ y f disminuye hacia la derecha.
- Para $x > 0$, $f'(x) > 0$ y f disminuye hacia la izquierda.
- Para $x = 0$ el gradiente se detiene puesto que $f'(x) = 0$.

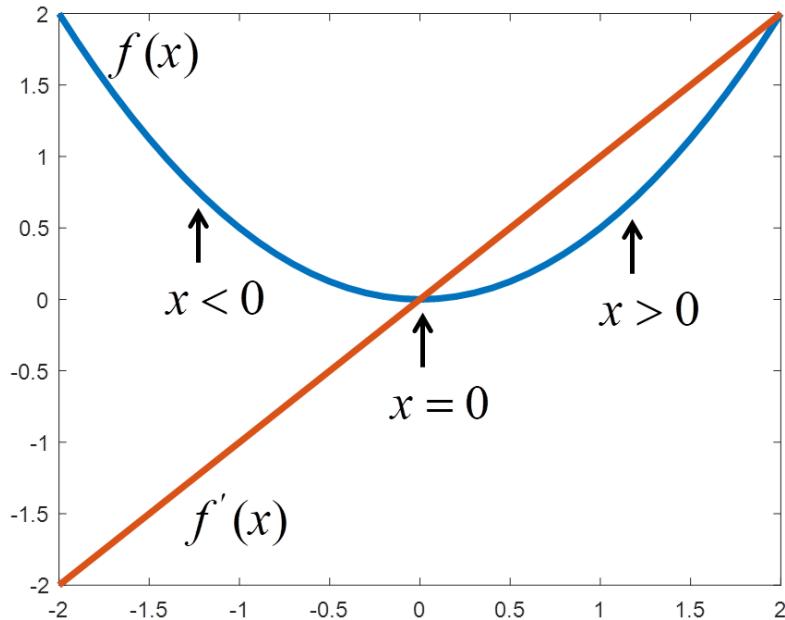


Figura 2.2: Representación gráfica del ejemplo.

Cuando la derivada vale 0 no proporciona ninguna información de la dirección en la que hay que moverse. De esta forma dichos puntos se conocen como puntos críticos o puntos estacionarios.

Un mínimo local es aquel punto donde $f(x)$ es menor que todos los puntos vecinos, de forma que no es posible disminuir $f(x)$ mediante pasos infinitesimales. Por otro lado, un mínimo global es aquel punto en el que $f(x)$ alcanza su valor más pequeño. Es normal que para una función existan muchos mínimos locales que no sean globales. Esto hace que la optimización sea compleja y no sea sencillo llegar a mínimos globales. En tales casos hay que conformarse con encontrar un valor muy bajo pero no necesariamente mínimo en un sentido formal, Figura 2.3



Figura 2.3: Puntos críticos.

2.3.1.1. Gradiente descendente estocástico

En el ámbito del aprendizaje automático, el problema del **Gradiente Descendente Estocástico** (SGD, *Stochastic Gradient Descent*) consiste en minimizar una función objetivo que tiene la forma (Bishop 2006; Ruder 2017).

$$J(w) = \frac{1}{n} \sum_{i=1}^n J_i(w) \quad (2.2)$$

donde el parámetro w , que minimiza $J(w)$, debe estimarse. J_i se asocia con la i -ésima observación en el conjunto de datos utilizados para el entrenamiento (ajuste). El término *estocástico* proviene del hecho relativo a la selección de las muestras (observaciones) para el ajuste de forma aleatoria, que incluso se pueden seleccionar por lotes en cada iteración. $J_i(w)$ es el valor de la *función de pérdida* (*loss function*) en el i -ésimo ejemplo y $J(w)$ es el riesgo empírico.

El gradiente descendente se usa para minimizar la siguiente función de forma iterativa, con iteraciones t :

$$w(t+1) = w(t) - \varepsilon \frac{1}{n} \sum_{i=1}^n \nabla_w J_i(w) \quad (2.3)$$

De esta forma iterativa, el método recorre el conjunto de entrenamiento y realiza la actualización indicada anteriormente para cada muestra de entrenamiento. Según la Ecuación (2.2), gradientes negativos incrementan el peso y viceversa, siempre con el fin de minimizar la función objetivo. Se pueden realizar varios pasos sobre el conjunto de entrenamiento hasta que el algoritmo converja. Si se hace esto, los datos se pueden seleccionar aleatoriamente en cada paso para evitar ciclos; es lo que se conoce como *criterio estocástico*. Estas muestras así seleccionadas constituyen lo que se denomina un *lote* (*batch*). Las implementaciones típicas pueden usar una razón de aprendizaje adaptativa para que el algoritmo converja. En pseudocódigo, el método del gradiente descendente estocástico es como sigue:

1. Elegir un vector inicial de **parámetros** w (puede ser aleatoriamente) y razón de aprendizaje, ε .

2. Repetir hasta que se consigue un mínimo aproximado:

- a) Seleccionar aleatoriamente ejemplos en el conjunto de entrenamiento.
- b) Para $i = 1, 2, \dots, n$, hacer

$$w(t+1) = w(t) - \varepsilon \nabla J_i(w) \quad (2.4)$$

Ejemplo: Supóngase que se quiere ajustar una línea recta $y = w_1 + w_2x$ a partir de un conjunto de observaciones (x_1, x_2, \dots, x_n) con sus correspondientes respuestas estimadas $(\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n)$, utilizando mínimos cuadrados. La función objetivo a minimizar es:

$$J(w) = \sum_{i=1}^n J_i(w) = \sum_{i=1}^n (\hat{y}_i - y_i)^2 = \sum_{i=1}^n (w_1 + w_2x_i - y_i)^2 \quad (2.5)$$

$$\begin{bmatrix} w_1(t+1) \\ w_2(t+1) \end{bmatrix} = \begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix} - \varepsilon \begin{bmatrix} \frac{\partial}{\partial w_1} (w_1 + w_2x_i - y_i)^2 \\ \frac{\partial}{\partial w_2} (w_1 + w_2x_i - y_i)^2 \end{bmatrix} = \begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix} - \varepsilon \begin{bmatrix} 2(w_1 + w_2x_i - y_i) \\ 2x_i(w_1 + w_2x_i - y_i) \end{bmatrix} \quad (2.6)$$

En cada iteración (también llamada actualización), solo se evalúa el gradiente en un único punto x_i en lugar de evaluar en el conjunto de todas las muestras. La diferencia clave en comparación con el gradiente descendente estándar es que solo se utiliza un dato del total del conjunto de datos disponible para obtener la actualización, y el dato se selecciona de forma aleatoria en cada paso.

2.3.1.2. SGD con Momentum

Existen diversas variantes, extensiones y mejoras del algoritmo. Resulta de particular interés el hecho de fijar la razón de aprendizaje, ya que valores altos tienden hacia la divergencia de los datos, mientras que valores demasiado pequeños hacen que la convergencia sea lenta. Una forma de abordar este problema es permitir que dicha razón sea variable, disminuyendo a medida que se incorporan nuevas muestras (mayor aprendizaje). Esto se consigue haciéndola dependiente del número de datos o iteraciones.

Una de tales extensiones es la del método del *momentum* o *momento* (Rumelhart y col.[9]; Murphy[10]). Este método recuerda la actualización en cada iteración, determinando la siguiente actualización como una combinación lineal del gradiente y la actualización previa (Sutskever y col.[11]):

$$\begin{aligned} \Delta \mathbf{w}(t) &= \alpha \Delta \mathbf{w}(t-1) - \varepsilon \nabla J_i(\mathbf{w}(t-1)) \\ \mathbf{w}(t) &= \mathbf{w}(t-1) + \Delta \mathbf{w}(t-1) \end{aligned} \quad (2.7)$$

La anterior expresión conduce a:

$$\mathbf{w}(t) = \mathbf{w}(t-1) + \alpha \Delta \mathbf{w}(t-1) - \varepsilon \nabla J_i(\mathbf{w}(t-1)) \quad (2.8)$$

donde $\epsilon < 0$ es la razón de aprendizaje, $\alpha \in [0, 1]$ es el momento constante que controla la velocidad de actualización de $\Delta\mathbf{w}(t-1)$. El vector de parámetros w , que minimiza $J(w)$, es el que se estima durante el proceso de optimización. El nombre de *momento* proviene del concepto de momento en física, de forma que el vector de pesos w , visto como una partícula viajando a través del espacio de parámetros, adquiere una aceleración a partir de la fuerza, tendiendo a mantenerse en la misma dirección y evitando así oscilaciones.

2.3.1.3. ADAM

ADAM, cuyo nombre deriva de *Adaptive Moment Estimation* (Kingma y Ba, 2015), utiliza un mecanismo de actualización de parámetros similar a RMSProp con momento. Mantiene una actualización de la media móvil, elemento a elemento, tanto de los gradientes de los parámetros como de sus valores al cuadrado.

$$\begin{aligned} m(t) &= \beta_1 m(t-1) + (1 - \beta_1) \nabla J_i(w(t-1)) \\ v(t) &= \beta_2 v(t-1) + (1 - \beta_2) (\nabla J_i(w(t-1)))^2 \\ w(t) &= w(t-1) - \frac{\alpha m(t-1)}{\sqrt{v(t-1)} + \epsilon} \end{aligned} \tag{2.9}$$

2.4. Funciones de activación

Las funciones de activación son componentes esenciales en las redes neuronales artificiales, introduciendo no linealidades que permiten a la red aprender relaciones complejas en los datos. Su elección afecta directamente a la capacidad de aprendizaje y eficiencia computacional del modelo.

2.4.1. Sigmoide

Una función de activación clásica es la función sigmoide o sigmoidal definida de la siguiente manera:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.10}$$

Dependiendo del signo del parámetro x , la función sigmoide se abre hacia la izquierda o hacia la derecha, siendo apropiada para representar conceptos tales como “muy grande” o “muy negativo”. La función sigmoide que proyecta salidas de números reales de entrada al intervalo $[0, 1]$ posee dos problemas:

- Saturación del gradiente. Cuando el valor de la función de activación se aproxima a los extremos 0 o 1, el gradiente de la función tiende a 0, lo que repercute en el ajuste de los pesos de las redes.

- Pesos positivos de forma continua. El valor medio de la función de salida no es 0, lo que origina que los pesos tiendan a ser positivos.

Estas dos cuestiones provocan una convergencia lenta de los parámetros afectando a la eficiencia del entrenamiento.

2.4.2. ReLU

La función Unidad Lineal Rectificada (ReLU, Rectified Linear Unit)

$$\text{ReLU}(x) = \max(0, x) \quad (2.11)$$

tiene las siguientes características:

- Gradiente no saturado. Por el hecho de que $x > 0$, el problema de la dispersión del gradiente en el proceso de propagación inversa se ve aliviado, y los parámetros en la primera capa de la red neuronal pueden actualizarse rápidamente.
- Baja complejidad computacional. Dada su propia definición. No obstante, posee la desventaja de que la neurona ReLU puede morir cuando recibe un gradiente negativo alto durante la retropropagación que le permite aprender más porque su derivada es cero cuando su entrada es menor que cero, por lo que el gradiente será finalmente cero.

Las funciones ReLU tienen la ventaja de acelerar el entrenamiento, ya que el cálculo del gradiente simple (0 o 1 dependiendo del signo de x). Además, el paso computacional de una unidad ReLU es fácil debido a que los elementos negativos se reducen a 0.0, sin exponentiales, y sin operaciones de multiplicación o división.

2.4.3. LeakyReLU

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{si } x > 0, \\ \alpha x & \text{si } x \leq 0 \end{cases} \quad (2.12)$$

Algunas de sus mejoras sobre ReLU son:

- Evita neuronas muertas: Al permitir un pequeño gradiente (α) para entradas negativas.
- Mantiene las ventajas de ReLU: Su eficiencia computacional y el gradiente constante para entradas positivas

2.4.4. SoftMax

La función softmax o función exponencial normalizada suele aparecer en las últimas capas ocultas. Se emplea para proyectar un vector n -dimensional, x de valores reales sobre un vector n -dimensional de valores reales en el rango $[0, 1]$. Calculándose como sigue:

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^n \exp(x_j)} \quad (2.13)$$

para $i = 1, \dots, n$ y

$$x = (x_1, x_2, \dots, x_n) \in i^n$$

2.5. Redes neuronales convolucionales (CNN)

Las CNN (*Convolutional Neural Networks*) son un tipo especializado de redes neuronales con una tipología basada en rejilla para el procesamiento de datos. Su aplicación se centra en el ámbito del sonido, imágenes y video, sobre los que se aplican operaciones basadas en rejillas de dimensiones 1-D, 2-D, 3-D o superiores, configuradas habitualmente como tensores, previamente definidos.

El término convolucional hace referencia a la operación matemática de convolución, operación lineal altamente especializada. Puede decirse que las CNN son redes neuronales que utilizan la convolución, en vez de la multiplicación de matrices, en al menos una de sus capas.

Cabe destacar que la operación de convolución suele ir acompañada de otras como puede ser la agrupación o pooling. Explicadas en posteriores apartados, las cuales aumentan el rendimiento de este tipo de redes y permiten operar con estas estructuras de datos de varias dimensiones.

A continuación se da una visión detallada de las operaciones y técnicas presentes en las estructuras de las CNN, siendo algunas de ellas exclusivas de este tipo de red.

2.5.1. Operación de convolución

La operación de convolución involucra dos funciones con valores reales como argumento. En el ámbito de las CNN el primer argumento x para la convolución se define como entrada (input) y el segundo w , como núcleo (kernel) de la convolución. La salida se define como un mapa de características (feature map). En el aprendizaje máquina la entrada es un vector o matriz multidimensional de datos, y el núcleo es, generalmente, un vector o matriz multidimensional de parámetros que se ajustan mediante el proceso de aprendizaje. Siendo ambas estructuras tensores. Por otra parte, las convoluciones son operaciones que se realizan sobre diferentes ejes al mismo tiempo. Esto sucede en el caso de las imágenes,

ya que son estructuras bidimensionales y por tanto necesitarán ser tratadas por un núcleo bidimensional K.

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n) \quad (2.14)$$

La operación de convolución es conmutativa, lo que significa que la expresión anterior puede ser escrita de forma equivalente de la siguiente manera, dando lugar a una forma más eficiente debido a una menor variación en los valores de m y n :

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n) \quad (2.15)$$

Otro factor relevante es lo que se conoce como correlación cruzada:

$$S(i, j) = (K \star I)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n) \quad (2.16)$$

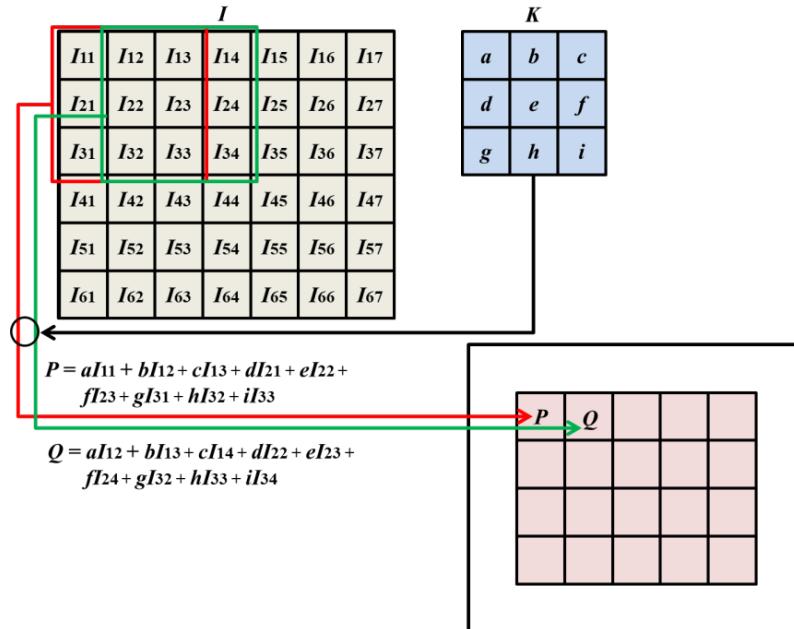


Figura 2.4: Ejemplo de convolución 2D.

La Figura 2.4 muestra un ejemplo de convolución con el núcleo K siendo aplicado sobre un tensor de dos dimensiones. Dicho tensor puede representar una imagen en blanco y negro (un solo valor por píxel). Cabe destacar que el tensor obtenido es de dimensión menor al actual; si se necesitase que se mantuvieran las dimensiones, sería suficiente con añadir filas y columnas de ceros a los lados, arriba y abajo. El número exacto es de sencillo cálculo. Esta técnica se denomina *zero-padding*. Estas operaciones pueden también ser aplicadas en una entrada de 3 dimensiones; esto implica que el núcleo fuese un cuboide que se desplaza a través de las dimensiones x, y, z del mapa de características. Este es el caso de las imágenes RGB, es decir, imágenes a color. Dichas imágenes presentan 3 canales para la capa de entrada, uno por cada color (rojo, verde y azul). Este proceso puede generalizarse para convoluciones de dimensión N.

De esta forma, los núcleos que definen una convolución discreta pueden definirse como una permutación de la siguiente forma: (n, m, K_1, \dots, K_n)

donde:

- n es el número de mapas de características de salida,
- m es el número de mapas de características de la entrada,
- K_j es la dimensión del núcleo a lo largo del eje j .

Cabe destacar que existen técnicas para variar los comportamientos de la convolución “clásica”, un ejemplo es el *stride*, que determina las unidades de las que consta el desplazamiento del núcleo. Por defecto, suele ser una unidad, pero dicho valor puede variar en función de las necesidades.

Un ejemplo ilustrativo es el siguiente. En el que hay una extensión de *padding* de 1 en ambas dimensiones (se rodea con una fila de ceros la entrada) y un *stride* en ambas dimensiones de 2 (el núcleo realiza saltos de tamaño dos)

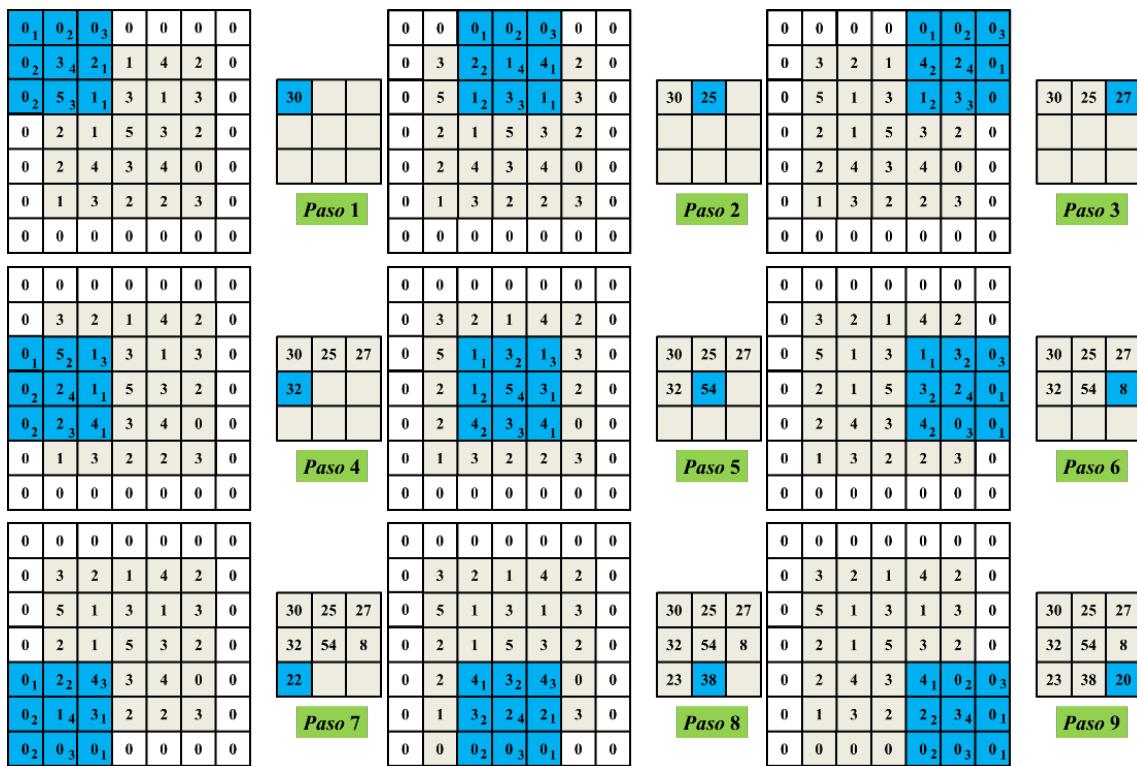


Figura 2.5: Convolución con padding y stride

2.5.2. Agrupamiento (Pooling)

Como fue mencionado con anterioridad una de las operaciones que se combinan con la convolución es la de agrupamiento. En las redes neuronales las capas denominadas de agrupamiento o *pooling*, proporcionan una importante invarianza a pequeñas traslaciones

de la entrada. Existen diferentes variantes, una de ellas es el máximo $\max()$, que consiste en dividir la entrada en ventanas, produciendo como salida el máximo de la ventana. Otro ejemplo es la media (mean)

Generalmente esta operación se aplica para reducir la dimensionalidad, de forma que la posición de la ventana se actualiza en función del desplazamiento (*strides*) previamente definido. Es importante recalcar que en ocasiones pueden quedar elementos, comúnmente los más alejados del centro, fuera de la ventana debido al tamaño de la misma, de la entrada y del *stride*. Por tanto para obtener los valores de las regiones borde debe realizarse una extensión como las vistas previamente (*zero-padding*). En la Figura 2.7 se observa una operación de *pooling* con un *stride* de la ventana de 2. En uno en el que no se emplea *padding* y por tanto quedan regiones sin aportar a los cálculos y otra en la que se rellena con *padding* de 1 en eje x y por tanto no quedan valores sin usar.

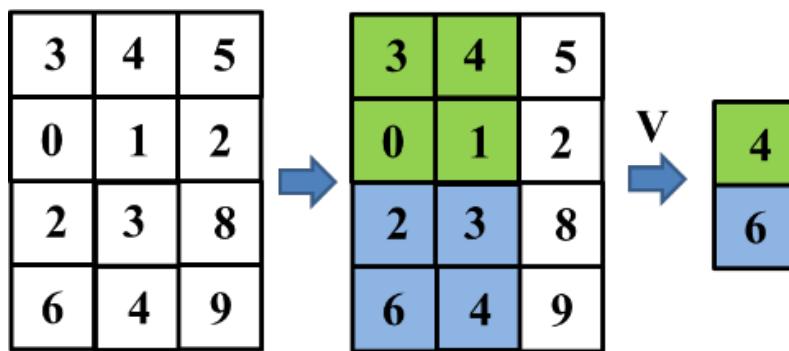


Figura 2.6: Pooling con elementos fuera de la ventana.

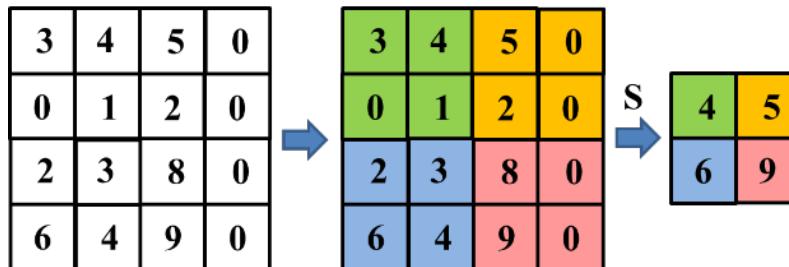


Figura 2.7: Pooling sin elementos fuera de ventana gracias a padding.

El *pooling* ayuda a hacer la representación aproximadamente invariante a pequeñas traslaciones de la entrada, lo que significa que si se traslada la entrada con un pequeño desplazamiento, los valores de muchas salidas sobre las que se ha aplicado el *pooling* no cambian.

Como se verá más adelante, una capa típica de convolución consta de tres estados. En el primer estado se realizan varias convoluciones para generar un conjunto de activaciones lineales. En el segundo estado, cada activación lineal se ejecuta a través de una función de activación no lineal, tal como la función de activación rectificada lineal (ReLU, Rectified Linear Unit). Este estado se conoce a veces como detector *stage*. En el tercer estado se utiliza la función de *pooling* para modificar la capa de salida, que tal y como se ha definido

previamente, reemplaza la salida de la red en una cierta localización con una operación estadística involucrando otras salidas cercanas.

2.5.3. Sobreajuste

Uno de los mayores problemas a la hora de evaluar estos modelos es lo que se conoce como sobreajuste u *overfitting*. En el caso de las redes neuronales este fenómeno ocurre cuando se ajustan un número elevado de pesos en numerosas neuronas. Esto puede provocar que el modelo se comporte de manera correcta con ciertos ejemplos pero sin embargo no sea capaz de generalizar bien y ante casos que el sistema nunca ha visto no se comporte de manera correcta. Podemos ver en la Figura 2.8 ambos extremos. En primer lugar vemos un ejemplo de *underfitting*, ya que el modelo pretende ajustarse a los puntos mediante una línea recta, es decir, no es capaz de ajustarse correctamente a los datos. Por otro lado encontramos el otro extremo, el *overfitting*, observamos que el ajuste se lleva a cabo mediante un polinomio de grado alto, es por eso que la línea roja presenta formas tan abruptas. Aunque el modelo se ajuste bien a dichos puntos no garantiza que vaya a generalizar bien ante nuevos datos. Es por eso que algo como lo que ocurre en la segunda figura es a lo que se intenta aspirar, un punto medio entre ambos casos

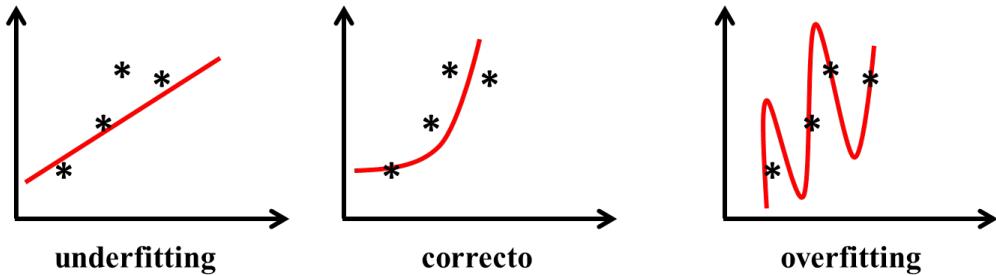


Figura 2.8: Diferentes comportamientos en función del ajuste.

Uno de los métodos para evitar el sobreajuste es aplicar una regularización, consistente en modificar la función objetivo a minimizar añadiendo términos adicionales que penalicen pesos con valores elevados. Por otro lado, otra técnica ampliamente utilizada es la de dividir los datos disponibles en tres conjuntos (aprendizaje, validación y test). De esta forma entrenamos nuestro modelo con el conjunto de aprendizaje, valoramos cuál es la mejor selección de hiperparámetros con el conjunto de validación y por último probamos el modelo óptimo con el conjunto de test. De esta forma nos aseguramos que se generan buenos resultados tanto para validación como para test.

Por último cabe destacar la detención temprana. Cuando un modelo se encuentra en proceso de entrenamiento es lógico que el error cometido sobre el conjunto de entrenamiento se vaya reduciendo con el paso el tiempo (*epochs*). Sin embargo no es conveniente prolongar el entrenamiento y mejorar los resultados en entrenamiento en exceso aunque pueda parecer contradictorio. Esto se debe a que llega un momento en el que el error en el conjunto de validación (que se debe observar a la par) deja de disminuir y aumentar, ya que nuestro

modelo comienza a ajustarse peligrosamente a los datos de entrenamiento y no adquiere la capacidad de generalización que requiere, puesto que el objetivo es obtener buenos resultados en datos que el modelo no ha visto nunca. Es por eso que determinar el punto de detección temprana puede evitar casos de sobreajuste, como se aprecia en la Figura 2.9.

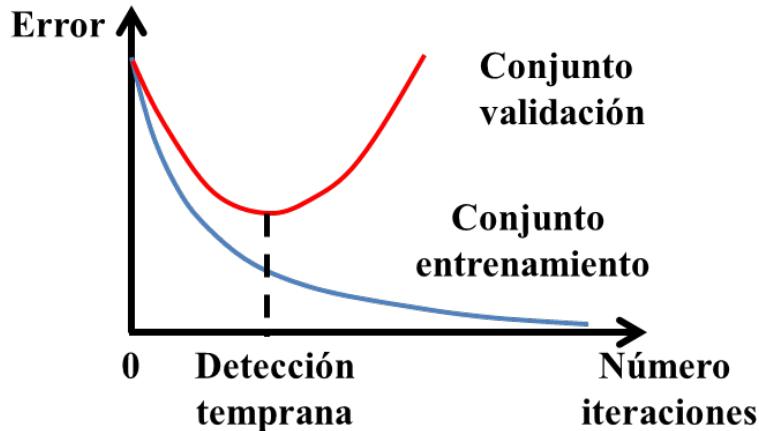


Figura 2.9: Detección temprana en error contra iteraciones.

2.5.4. Normalización

La operación de normalización destaca por mejorar el rendimiento de las redes neuronales profundas. Permite acelerar la convergencia y que se generalice mejor. Se denota por $a_{x,y}^i$ la actividad de una neurona obtenida por aplicación del núcleo i en la posición (x, y) y luego se aplica la no linealidad ReLU, la actividad de la respuesta normalizada $b_{x,y}^i$ viene dada por la expresión,

$$b_{x,y}^i = a_{x,y}^i \left/ \left[k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^j)^2 \right]^\beta \right. \quad (2.17)$$

donde N es el número de filtros de la capa dada, $a_{x,y}^i$ es la actividad de la neurona y , α , n y β son hiperparámetros. En Krizhevsky[12] se proponen como valores adecuados para dichos parámetros $k = 2$, $n = 5$, $\alpha = 10^{-4}$, $\beta = 0,75$.

Durante el entrenamiento en las redes neuronales profundas la distribución de las entradas de cada capa no es fija, es decir, cambia en función de los comportamientos de las capas anteriores. Esto implica que se ralentice el proceso, ya que se requieren tasas de aprendizaje más bajas y una cuidadosa selección inicial de los parámetros. Esto se denomina desplazamiento covariante interno (*covariate shift*). Este problema se aborda normalizando las capas de entrada, incorporando esta acción al diseño del modelo.

La normalización se realiza sobre cada mini-lote (*mini-batch*) de entrenamiento, permitiendo emplear razones de aprendizaje más altas y permitiendo tener menos cuidado con la inicialización de parámetros.

2.5.5. Mean Squared Error

El error cuadrático medio (MSE, mean square error), también denominado en el contexto del aprendizaje como quadratic loss o L2 loss, se define como,

$$ECM = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.18)$$

donde n es el conjunto de muestras de entrenamiento, y los términos y_i e \hat{y}_i se refieren, respectivamente, al valor deseado y al predicho.

2.5.6. Cross-Entropy Loss

Utilizada generalmente en clasificación, la entropía cruzada (*cross-entropy*), también conocida como *log loss*, determina el desempeño del modelo de clasificación cuya salida es un valor de probabilidad entre 0 y 1. El valor de la función de entropía cruzada aumenta a medida que la probabilidad predicha diverge de la etiqueta actual.

Así, por ejemplo, si se predice una probabilidad de 0,1 cuando la etiqueta de la observación es 1, el resultado es malo, con un alto valor de pérdida. Se define como sigue en términos de probabilidad:

$$H(p, q) = - \sum_x p(x) \log q(x) \quad (2.19)$$

2.6. Detección de objetos en imágenes

2.6.1. Introducción

La idea subyacente cuando se habla de la detección de objetos en imágenes es la de determinar dentro de la misma las zonas de interés (regiones), que se recortan y se pasan a la CNN para su clasificación, obteniendo una probabilidad de que dicha región contenga un objeto de una determinada clase.

La estructura que presentan este tipo de estrategias es la siguiente: una columna vertebral o red troncal y una cabeza encargada de predecir las clases además de los *bounding boxes*. Para la columna vertebral se emplean distintos tipos de detectores, algunos son VGG, ResNet o DenseNet. En el caso de la cabeza se dividen en dos categorías: detector de un estado y de dos estados. En el caso de los primeros, estos destacan por procesar la imagen completa una sola vez mientras que en el segundo caso se procesan las regiones por separado. Algunos ejemplos de dos estados son: R-CNN y sus variantes R-FCN. Por el lado de los de un estado destacan SDD y YOLO, modelo empleado en la realización de este proyecto y que será presentado con detalle en posteriores apartados. Algunos detectores

a menudo insertan algunas capas entre la columna y la cabeza para recopilar mapas de características procedentes de diferentes etapas. Suelen denominarse cuello de un detector. En la Figura 2.10 se muestra la estructura general de los detectores.

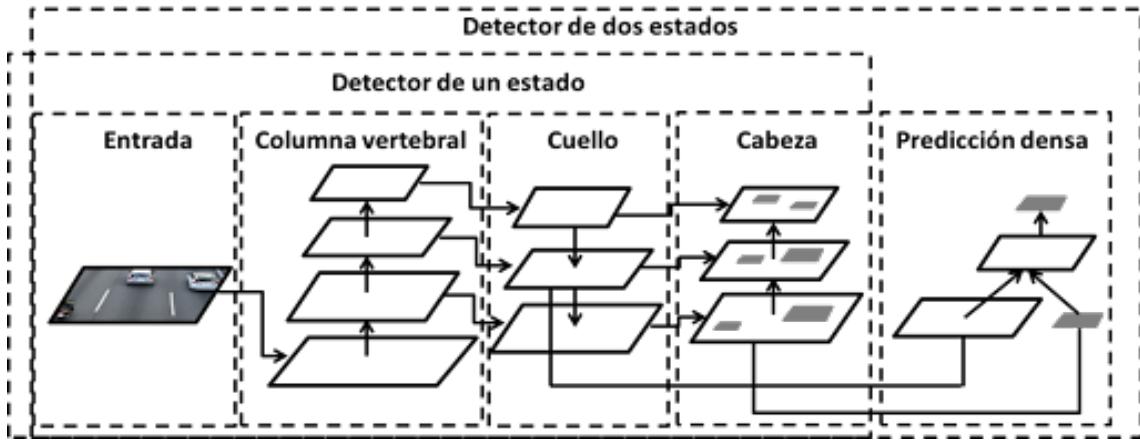


Figura 2.10: Arquitectura general de la detección de objetos

2.6.1.1. Solapamiento de regiones y precisión

El poder detectar el solapamiento de regiones es imprescindible para la detección de objetos. El solapamiento se mide comúnmente con el índice de Intersección sobre la Unión (IoU), Figura 2.11, el cual permite al algoritmo ver la similitud entre una *bounding box* propuesta y un *ground truth* (el objeto real). Para evaluar un *bounding box* propuesto tenemos su clase, su área (*bounding box*) y su puntuación de confianza (confidence score) P_0 . Para determinar si una detección es válida, un VP (verdadero positivo) tiene que cumplir tres condiciones: que el P_0 sea mayor a cierto umbral, que la clase predicha sea la misma que la clase del *ground truth* y por último, el IoU tiene que ser mayor a otro umbral. Este índice se calcula dividiendo el área de solapamiento por el área de unión de los dos objetos. Si incumple una de las 2 últimas, se considera un FP (falso positivo) y si la P_0 es menor de su umbral pero si está marcando un *ground truth* se considera un FN (falso negativo) y, finalmente, si no debería detectar nada y su P_0 es menor, es decir, que es correcto el que no debe detectar nada, se considera un VN (verdadero negativo).

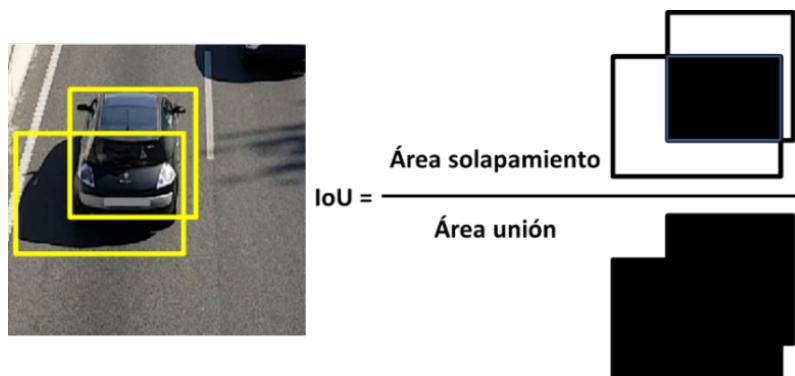


Figura 2.11: Intersección sobre la Unión formula

2.6.1.2. *Anchor boxes*

En el ámbito de la detección de objetos, los *anchor boxes* son regiones predefinidas dentro de una imagen que sirven para facilitar la identificación y localización de objetos. Estos rectángulos actúan como referencias que determinan si una determinada zona contiene un objeto de interés. Su función es permitir que la red ajuste estas regiones hasta alinearlas con los *bounding boxes* del objeto real (*ground truth*), con el fin de delimitar este de la forma más precisa posible. La principal diferencia entre ambos radica en que los *anchor boxes* representan propuestas de regiones iniciales, mientras que los *bounding boxes* finales son refinados a partir de estas. Para generar *anchor boxes*, hay diferentes estrategias que generan múltiples rectángulos con distintas dimensiones y relaciones de aspecto en cada píxel de la imagen. De esta forma, se pueden definir cuadrados o rectángulos de diversas escalas que cubran todos los objetos de diferentes tamaños y orientaciones. Por ejemplo, si una imagen tiene dimensiones de 5300×3000 píxeles, se pueden establecer *anchor boxes* con relaciones de aspecto de 1:1, 0.5:1 y 2:1, ajustando su escala (s) en función del tamaño global de la imagen, y en el caso de que se salga de la imagen solo se considera la parte del rectángulo que se queda dentro de ella, Figuras 2.12–2.14



Figura 2.12: $s_0 = 1$

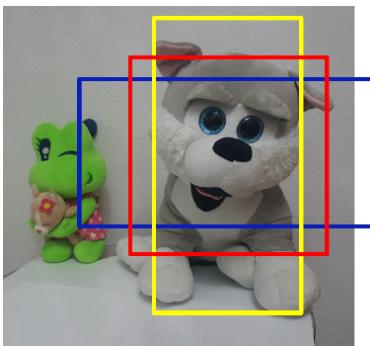


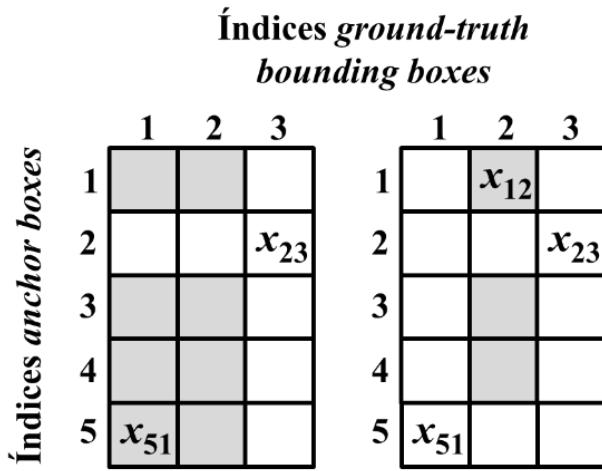
Figura 2.13: $s_1 = 2$



Figura 2.14: $s_2 = 0,5$

Una vez generados, los *anchor boxes* deben asociarse a los *bounding boxes* reales mediante la métrica IoU (Intersection over Union), que mide el grado de solapamiento entre ambos. Esto se hace con una matriz donde A_1, A_2, \dots, A_n son los *anchor boxes* y B_1, B_2, \dots, B_m son los *ground truth bounding boxes* con $m \leq n$. Se define la matriz donde el elemento x_{ij} en la i -ésima fila y j -ésima columna es el IoU del *anchor box* A_i al *ground truth bounding box* B_j . Una vez creada la matriz, el proceso de asociación comienza de manera iterativa hasta quedarse sin *ground truth* para escoger. Este empieza cogiendo el valor IoU más alto y asocia ese *anchor box* al *ground truth* siempre que pase de un cierto umbral determinado y descarta esa fila y esa columna para que no se repita. De esta manera quedan solo los mejores candidatos y correctamente vinculados.

Según la Figura 2.15, este sería un ejemplo en el que el mayor X_{ij} es el X_{23} , descartando así la columna 3 y la fila 2, y en el segundo paso el mayor índice sería X_{51} , descartando la columna 1 y la fila 5, y finalmente se seleccionará X_{12} como el mayor elemento y se descartará la columna 2 y la fila 1.

Figura 2.15: Indices *Ground truth*

Cada *anchor box* recibe una etiqueta basada en esta asignación. Si está asociado a un *bounding box* real, se clasifica como positivo y se le asigna la categoría del objeto correspondiente. En cambio, si no tiene una asociación válida, se clasifica como negativo, indicando que pertenece al fondo de la imagen. Además de la categoría, los *anchor boxes* etiquetados también contienen información sobre el desplazamiento necesario para ajustarse mejor a la ubicación y dimensiones del objeto.

Cuando existen muchos *anchor boxes*, es posible que se realicen predicciones muy similares en relación con los *bounding boxes* para un mismo objeto. Para evitar duplicaciones y mejorar la precisión, se emplea el método de *supresión no-máxima* (NMS, *Non-Maximum Suppression*), cuyo objetivo es eliminar predicciones redundantes y conservar únicamente aquellas relevantes.

El proceso NMS tiene los siguientes pasos:

1. Primero, se ordenan todas las predicciones de los *bounding boxes* según su nivel de confianza, sin incluir aquellos clasificados como fondo.
2. En segundo lugar, se selecciona el *bounding box* con la mayor confianza y se establece como referencia.
3. Finalmente, se eliminan todos los *bounding boxes* con un IoU superior a un umbral predefinido con respecto a este *bounding box* de referencia. El proceso se repite con el segundo *bounding box* con mayor confianza, eliminando nuevamente aquellos con IoU elevado. Esta iteración continúa hasta que se han procesado todos los *bounding boxes*, garantizando que los resultados finales incluyan únicamente los más representativos.

Este sería un ejemplo del NMS, Figuras 2.16 y 2.17 en la imagen izquierda, se pueden ver múltiples *bounding boxes* predichos para dos objetos, cada uno con un nivel de confianza

(p) asociado. Para el Peluche-2, existen tres cajas con $p = 0,9$ (verde), $p = 0,7$ (rojo) y $p = 0,6$ (amarillo), mientras que para el Peluche-1, hay una única caja amarilla con $p = 0,8$.

El proceso NMS comienza ordenando en una lista los *bounding boxes* de mayor a menor confianza, seleccionando primero la caja con $p = 0,9$ como referencia. Luego, se eliminan aquellas cajas con un IoU superior a un umbral predefinido (0.5 en este caso), eliminando así las cajas roja y amarilla de Peluche-2 por solaparse demasiado con la verde. Como resultado, en la imagen derecha solo se conservan las cajas más representativas: la verde con $p = 0,9$ para Peluche-2 y la amarilla con $p = 0,8$ para Peluche-1, habiendo eliminado las predicciones redundantes.

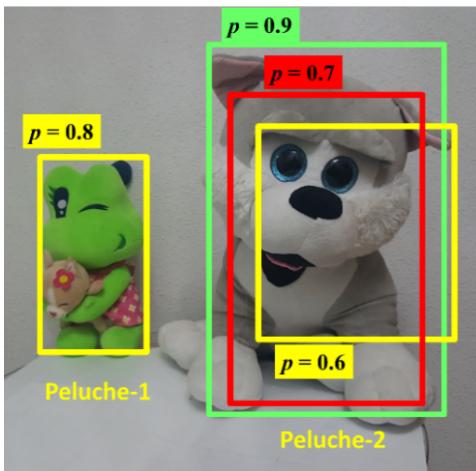


Figura 2.16: *Bounding Boxes* solapadas

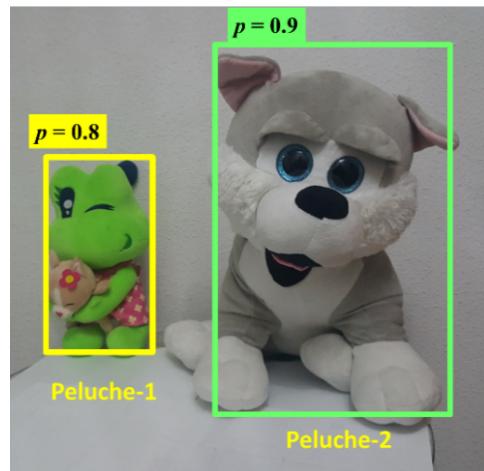


Figura 2.17: *Bounding Boxes* finales

2.6.2. YOLO

2.6.2.1. Introducción

YOLO (You Only Look Once)[13] es un algoritmo de detección de objetos en tiempo real que revolucionó el campo de la visión por computadora al ofrecer un enfoque unificado y en tiempo real para la detección de objetos. A diferencia de los métodos que en su momento eran tradicionales, como R-CNN, que utilizaba region proposals (propuestas de regiones) en múltiples etapas, YOLO aborda la detección de objetos como un problema de regresión. Esto significa que predice las coordenadas de las *bounding boxes* y las clases de objetos en una sola pasada a través de la red neuronal, lo que lo hace extremadamente rápido y eficiente. A continuación se presentan las características principales de YOLO, que han permitido a dicha familia diferenciarse de otras ya existentes.

- **Velocidad:** YOLO es mucho más rápido que otros métodos de detección de objetos, debido al enfoque empleado para resolver el problema, que ha supuesto una innovación en el campo de los detectores de objetos de un estado.

- Precisión: Aunque en las primeras versiones sacrificaba un poco de precisión a cambio de velocidad, en las versiones posteriores se mejoró mucho en ese aspecto.
- Enfoque de regresión: Predice *bounding boxes* y clases en una sola pasada.
- Dificultad con objetos pequeños: YOLO puede tener problemas para detectar objetos pequeños o muy cercanos entre sí.

Con respecto a la evolución de YOLO, cabe mencionar que ha evolucionado significativamente desde su primera versión en 2015, de la cual se hablará más adelante. Cada nueva versión ha introducido mejoras en términos de precisión, velocidad y eficiencia. En los siguientes puntos se tratarán las versiones que se han utilizado en el proyecto, las cuales son algunas de las versiones más destacadas. No obstante, primero se exponen algunas de las aplicaciones y sectores en los que esta familia ha contribuido con avances significativos.

YOLO se utiliza en una amplia variedad de aplicaciones en nuestra vida cotidiana. Algunos de los ámbitos en los que más se utilizan estas aplicaciones son: sanidad, agricultura, vigilancia de la seguridad y coches autónomos.

- Sanidad: Concretamente en cirugía, puede ser un reto localizar órganos en tiempo real, debido a la diversidad biológica de un paciente a otro.
- Agricultura: Implementado sobre todo en robots recolectores, los cuales son robots basados en la visión que se introdujeron para sustituir la recolección manual de frutas y verduras. También facilitó la detección de plagas en los cultivos.
- Vigilancia de seguridad: Aunque la detección de objetos se utiliza sobre todo en la vigilancia de seguridad, no es la única aplicación. Durante la pandemia de Covid19 se utilizó YOLO para estimar las violaciones de la distancia social entre las personas.
- Coches autónomos: La detección de objetos en tiempo real forma parte del ADN de los sistemas de vehículos autónomos. Esta integración es vital para los vehículos autónomos, porque necesitan identificar correctamente los carriles correctos y todos los objetos y peatones circundantes para aumentar la seguridad vial.

2.6.2.2. Primera aproximación

YOLOv1[14], introducido en 2016, fue la primera versión del algoritmo YOLO. Propuso un enfoque innovador el modelo de detección se plantea como un problema de regresión, de forma que la imagen se divide en una rejilla de dimensión $S \times S$ celdas y para cada celda de la rejilla se predicen B *bounding boxes*, junto con los valores de confianza de clase para esos boxes y las C probabilidades condicionadas de pertenencia a una clase determinada para un objeto dado. Esas predicciones se codifican como un tensor de dimensión $S \times S \times (5B + C)$. Respecto al diseño de la red, en el trabajo de Redmon y col. (2016) se propone un modelo de arquitectura que consta de 24 capas convolucionales seguidas por 2 totalmente conectadas. Se usan capas de reducción 1×1 seguidas por capas de convolución de dimensión 3×3 . De esta forma, alternando capas de convolución 1×1 se reducen las dimensiones del espacio

de características a partir de las capas precedentes. La red completa se muestra en la Figura 2.18, produciendo un tensor a la salida de dimensión $7 \times 7 \times 30$.

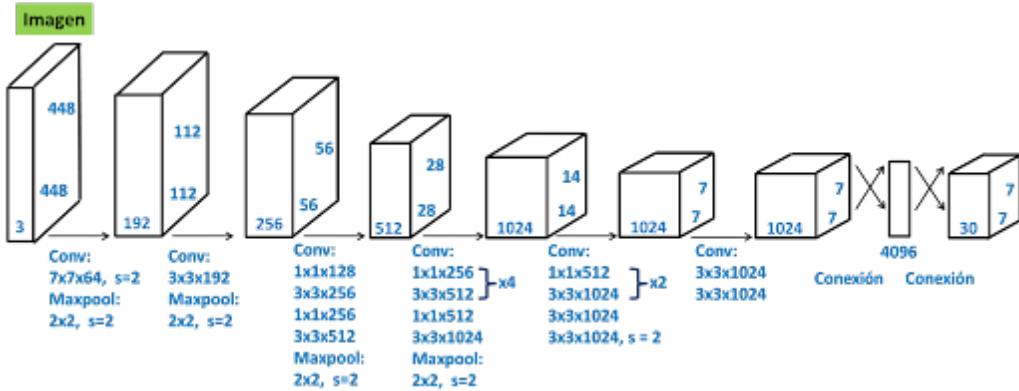


Figura 2.18: Capas red Yolo.

Para calcular la pérdida durante el entrenamiento, es necesario que solo uno de estos *bounding boxes* sea el “responsable” de detectar el objeto. Para ello, se selecciona el *bounding box* que tenga el mayor IoU (*Intersection over Union*) con el *ground truth* (la anotación real del objeto). Esta selección permite que cada *bounding box* se especialice en predecir ciertas características, con el tamaño o la forma de los objetos, mejorando progresivamente su precisión. *YOLO* utiliza el error cuadrático para medir la diferencia entre las predicciones del modelo y los valores reales (*ground truth*). La función de pérdida se divide en tres componentes principales:

Pérdida de clasificación

La pérdida de clasificación se calcula solo en las celdas donde se detecta un objeto y mide el error en la probabilidad de cada clase:

$$\sum_{i=0}^{S^2} O_i^{obj} \sum_{c \in \text{clases}} (p_i(c) - \hat{p}_i(c))^2 \quad (2.20)$$

donde: $O_i^{obj} = 1$ si un objeto aparece en la celda i , de otro modo es 0; $p_i(c)$ denota la probabilidad de clase condicionada para la clase c en la celda i . $\hat{p}_i(c)$ es la probabilidad predicha por el modelo.

Pérdida de localización

La pérdida de localización mide los errores en la predicción de las coordenadas y dimensiones de los *bounding boxes*:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B O_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \quad (2.21)$$

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B O_{ij}^{obj} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \quad (2.21)$$

donde: λ_{coord} es un factor de peso que aumenta la importancia de la pérdida en la localización. (x_i, y_i) y (\hat{x}_i, \hat{y}_i) son las coordenadas reales y predichas del centro del *bounding box*. (w_i, h_i) y (\hat{w}_i, \hat{h}_i) son el ancho y la altura reales y predichos del *bounding box*. La raíz cuadrada se usa en w y h para reducir el impacto de los errores en cajas grandes.

Pérdida de confianza

La pérdida de confianza mide qué tan bien el modelo predice la presencia de un objeto en una celda dada. Se divide en dos partes: cuando hay un objeto presente y cuando no lo hay.

Si hay un objeto en la celda:

$$\sum_{i=0}^{S^2} \sum_{j=0}^B O_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 \quad (2.22)$$

Si no hay un objeto en la celda:

$$\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B O_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2 \quad (2.23)$$

donde: C_i es la confianza real (IoU entre la predicción y el *ground truth*). \hat{C}_i es la confianza predicha. λ_{noobj} reduce la penalización de falsas detecciones en celdas sin objetos.

Finalmente, la función de pérdida total en *YOLO* se define como:

$$\text{Loss} = \text{Pérdida de Localización} + \text{Pérdida de Confianza} + \text{Pérdida de Clasificación}$$

2.6.2.3. YOLOv4

YOLOv4[15], propuesto por Bochkovskiy et al. (2020), sigue el esquema general de los detectores de objetos y se compone de tres partes principales: Columna vertebral (*backbone*), utiliza una versión modificada de DarkNet53, que es una red convolucional eficiente para la extracción de características, Cuello (*neck*), emplea técnicas como *Spatial Pyramid Pooling* (SPP) o *Path Aggregation Network* (PaNet) para mejorar la fusión de características a diferentes escalas y Cabecera (*head*), utiliza la misma cabecera que *YOLOv3*, lo que permite mantener la eficiencia en la detección de objetos.

Introduce técnicas de data augmentation (aumento de datos) durante el entrenamiento, conocidas como *bag of freebies*, que mejoran la precisión del modelo sin incrementar el coste computacional durante la inferencia. Estas técnicas incluyen transformaciones de imágenes que aumentan la diversidad del conjunto de datos y métodos para equilibrar la pérdida y mejorar la generalización del modelo.

También se enfoca en reducir el coste computacional durante la inferencia, manteniendo una alta precisión. Esto se logra mediante técnicas conocidas como *bag of specials*, que incluyen mecanismos para optimizar el procesamiento de características y estrategias para mejorar la eficiencia sin sacrificar el rendimiento.

2.6.2.4. *YOLOX*

YOLOX[16] es una de las versiones más modernas de la familia de detectores. Es un modelo sin anclajes (*anchor-free*), que mejora significativamente la eficiencia y velocidad en comparación con versiones anteriores de *YOLO*. A diferencia de los modelos *YOLO* clásicos, *YOLOX* no utiliza *anchor boxes* predefinidos, lo que reduce el tamaño del modelo y simplifica el proceso de detección. En su lugar, *YOLOX* localiza objetos directamente encontrando sus centros y predice las dimensiones de las cajas delimitadoras (*bounding boxes*) dividiendo la imagen en una cuadrícula de tres escalas diferentes. Este enfoque permite realizar entrenamiento basado en mosaicos (*tile-based training*), donde la red se entrena en parches de la imagen y se infiere en imágenes completas. La arquitectura de *YOLOX* consta de tres componentes principales:

Backbone

Utiliza una red convolucional preentrenada llamada CSP-DarkNet-53, entrenada en el conjunto de datos COCO[17]. Esta red actúa como extractor de características, generando mapas de características a partir de las imágenes de entrada.

Neck

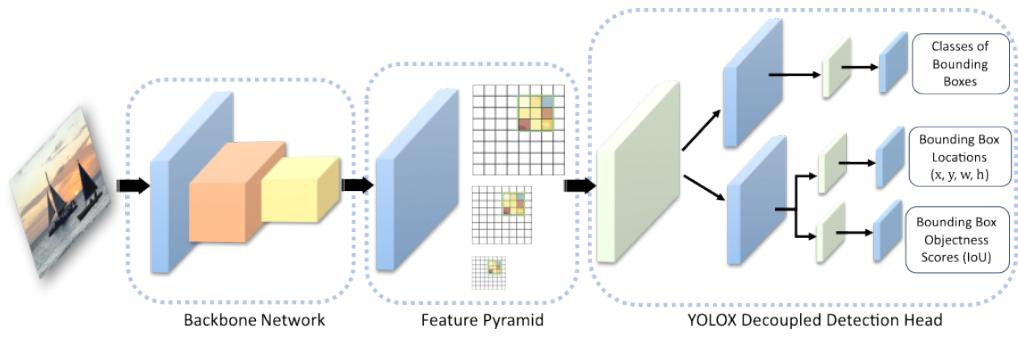
Conecta el *backbone* con el *head* y está compuesto por una Red de Pirámide de Características (FPN) y una Red de Agregación de Rutas (PAN). La FPN genera mapas de características a múltiples escalas, mientras que la PAN combina características de bajo y alto nivel. El *neck* envía estas características al *head* en tres escalas diferentes (1024, 512 y 256 canales).

Head

La cabeza de detección desacoplada (*decoupled detection head*) procesa las características agregadas y genera tres salidas:

- Puntuaciones de clasificación: Indican la clase de cada objeto detectado.
- Puntuaciones de regresión: Determinan la ubicación y dimensiones de las cajas delimitadoras
- Puntuaciones de objetividad (IoU): Miden la confianza de que una caja delimitadora contiene un objeto.

Esta arquitectura permite a *YOLOX* ser eficiente en términos de computación y precisión, lo que lo hace adecuado para aplicaciones en tiempo real.

Figura 2.19: Arquitectura red *YoloX*.[1]

Capítulo 3

Diseño y análisis de la aplicación: descripción del trabajo

Una vez se ha realizado el estudio de los conceptos teóricos subyacentes a los modelos, se continúa especificando el proceso de creación de una aplicación capaz de emplear dichas técnicas. Detallando la arquitectura de la misma, las distintas funcionalidades disponibles, así como una descripción de la gestión del proyecto. Desde un punto de vista técnico, o de desarrollo desde un factor humano, incluyendo la organización y comunicación entre los miembros.

3.1. Arquitectura

3.1.1. Capas de la aplicación

La arquitectura de la aplicación está basada en una *estructura cliente-servidor*¹ dividida en dos capas:

- Capa de presentación (*Front-End*) Capa que será visible por parte del cliente. Presenta una interfaz gráfica con la que el usuario debe interactuar para realizar entrenamientos y clasificaciones de manera sencilla, así como recibir información acerca de procesos ejecutados en la capa de negocio.

¹La *estructura cliente-servidor* es un modelo de comunicación en el que un cliente realiza peticiones a un servidor, el cual proporciona respuestas o servicios. Es ampliamente utilizada en aplicaciones web y sistemas distribuidos.

- Capa de negocio (*Back-End*) Capa que contiene la lógica subyacente a las decisiones del cliente, además de los mecanismos necesarios para llevar a cabo las tareas seleccionadas y enviar resultados a la capa de presentación para que el usuario pueda recibir información. Gestiona la carga y procesamiento de imágenes, así como el almacenamiento y recuperación de modelos entrenados.

3.2. Casos de uso

El diagrama de casos de uso, Figura 3.1, es una representación visual que muestra las interacciones entre un usuario (actor) y la aplicación. De este modo se permite obtener una visión general de las funcionalidades al alcance del usuario.

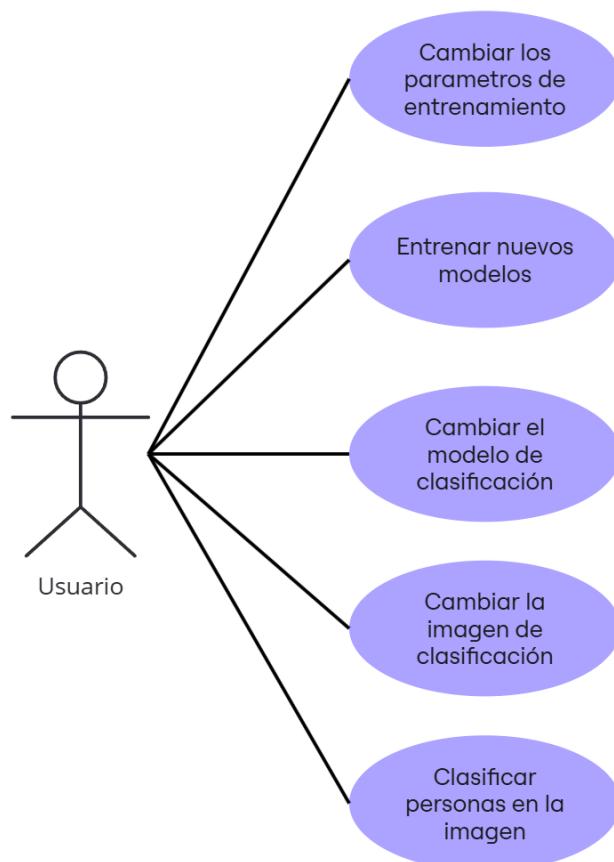


Figura 3.1: Diagrama de casos de uso

A continuación se describen cada uno de sus componentes:

- **Seleccionar/Ajustar los parámetros de entrenamiento:** Permite al usuario modificar valores que controlan el proceso de entrenamiento del modelo. Esto incluye la posibilidad de cambiar el propio modelo *YOLO*, seleccionar un optimizador distinto o ajustar parámetros específicos como *MaxEpoch*, *initialLearnRate*, *MiniBatchSize*, *VerboseFrequency* y *ValidationFrequency*. Dichos hiperparámetros serán definidos con propiedad a continuación. Esta funcionalidad dota al usuario de una amplia variedad de configuraciones con las que posteriormente entrenar.
- **Entrenar nuevos modelos:** Permite al usuario iniciar el entrenamiento de un modelo de clasificación con los parámetros establecidos, utilizando el conjunto de entrenamiento.
- **Cambiar el modelo de clasificación:** El usuario puede seleccionar entre distintos modelos de clasificación previamente entrenados o seleccionar uno que haya entrenado él mismo desde la aplicación, eligiendo el más adecuado para su tarea.
- **Cambiar la imagen de clasificación:** Permite al usuario subir una nueva imagen propia que se utilizará para probar el modelo de clasificación.
- **Clasificar personas en la imagen:** Ejecuta el modelo de clasificación sobre la imagen cargada para identificar y clasificar a las personas en ella.

3.3. Inicio de la aplicación

Todas las funcionalidades anteriores han sido integradas dentro de la aplicación. En primer lugar, cuando se ejecuta la aplicación, el usuario se encuentra con la siguiente ventana Figura 3.2. Al *hacer click* sobre el botón *start* se avanza al menú principal de la aplicación, Figura 3.3, en el que se le da al usuario la opción de elegir entre las dos funcionalidades principales de la aplicación: entrenar y clasificar. A continuación, se explican con todo lujo de detalle ambos apartados, ya que presentan una complejidad superior.



Figura 3.2: Ventana inicial de la aplicación.



Figura 3.3: Menú principal con las opciones de entrenamiento y clasificación.

3.4. Fase de entrenamiento

Como se ha mencionado anteriormente, una de las funcionalidades fundamentales de la aplicación es la de entrenar nuevos modelos de *YOLOv4* y de *YOLOX*, de forma que el usuario pueda elegir libremente la configuración del entrenamiento configurando ciertos hiperparámetros.

El dataset con las imágenes y anotaciones es esencial para el entrenamiento de los modelos. Sin embargo, no es necesario que el dataset esté en una carpeta específica predefinida por la aplicación. En su lugar, la aplicación permite al usuario seleccionar el directorio donde se encuentran las imágenes y anotaciones durante el proceso de entrenamiento.

3.4.1. Modelos, optimizadores e hiperparámetros

- **Modelos** La primera elección que el usuario debe realizar es la de decidir si su entrenamiento será un modelo *YOLOv4* o *YOLOX*.
- **Métodos de optimización** A continuación, el usuario debe seleccionar el método de optimización que se utilizará durante el entrenamiento. Las opciones disponibles son:
 - **ADAM** (*Adaptive Moment Estimation*): Un optimizador adaptativo que ajusta la tasa de aprendizaje durante el entrenamiento.
 - **SGDM** (*Stochastic Gradient Descent with Momentum*): Un optimizador clásico que utiliza el momento para acelerar la convergencia y evitar oscilaciones en el descenso del gradiente.

- **MaxEpochs** Valor predeterminado 30 Parámetro que determina el número máximo de épocas (pasos completos de los datos) que se desea emplear en el entrenamiento, es decir, el número máximo de veces que el modelo verá todo el conjunto de datos durante el entrenamiento. Debe ser un entero positivo. Un valor alto puede mejorar la precisión pero aumenta el tiempo de entrenamiento y el riesgo de sobreajuste
- **InitialLearningRate** Valor predeterminado en ADAM 0.001 y 0.01 en SGDM. Es la tasa de aprendizaje inicial usada para el entrenamiento. Debe ser un escalar positivo. Cabe destacar que si dicho valor es demasiado bajo, el entrenamiento puede prolongarse mucho en el tiempo. Y por el contrario si es demasiado alto, el entrenamiento podría no conseguir un resultado óptimo o divergir. Este parámetro tan solo es seleccionable cuando el optimizador es estocástico, como es el caso de “ADAM” y “SGDM”
- **MiniBatchSize** Valor predeterminado 128 Es el tamaño del minilote que se desea usar para cada iteración del entrenamiento. Comprendido como un entero positivo. Un mini batch es un subconjunto del conjunto de entrenamiento que se usa para evaluar el gradiente de la función de pérdida y por consiguiente actualizar los pesos. Un tamaño de minilote grande puede acelerar el entrenamiento, pero requiere de más memoria.
- **VerboseFrequency** Valor predeterminado 50 Frecuencia de la impresión detallada, que es el número de iteraciones entre cada impresión en la ventana de comandos, especificada como un entero positivo.
- **ValidationFrequency** Valor predeterminado 50 Frecuencia de la validación de la red neuronal en número de iteraciones, especificado como un entero positivo

De esta forma se presenta al usuario la funcionalidad de elección de parámetros:

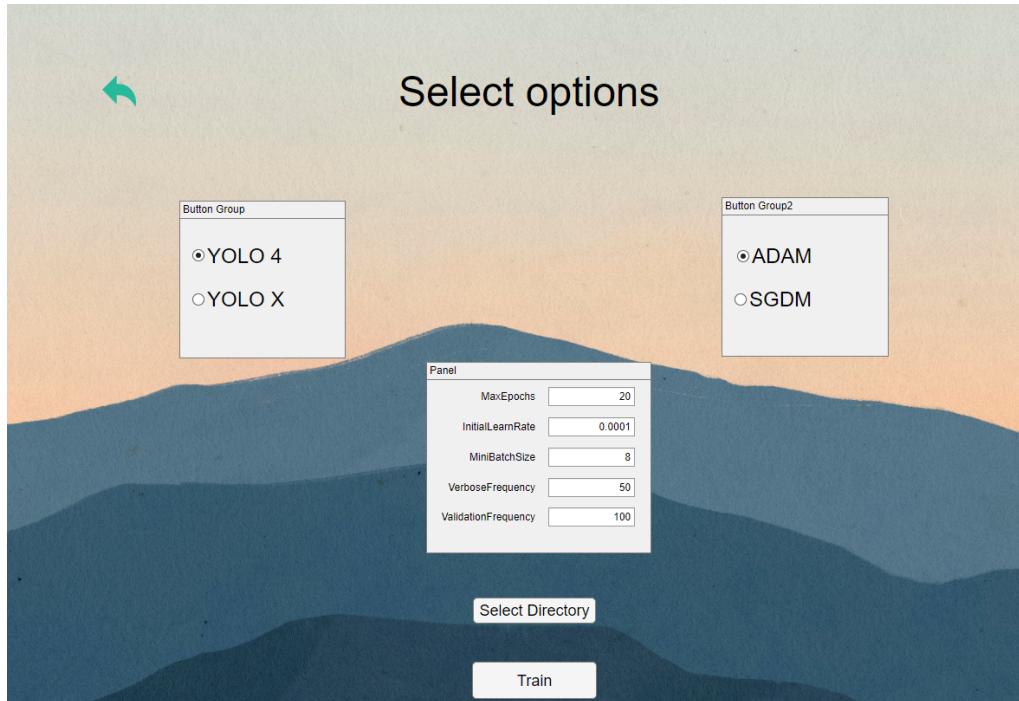


Figura 3.4: Selección de parámetros de entrenamiento.

3.5. Fase de clasificación

La segunda funcionalidad principal de la aplicación es la de clasificar una imagen. Entendemos por clasificar el proveer a un modelo seleccionado de una imagen, modificar una serie de opciones y obtener una imagen de salida en la que se generen los *bounding boxes* oportunos en las siluetas que el modelo ha considerado humanas.

3.5.1. Carga de imágenes

La aplicación incluye una imagen predeterminada que se muestra al iniciar la fase de clasificación. Esto permite al usuario probar la funcionalidad sin necesidad de cargar una imagen. Pero también se da la opción de que el usuario pueda cargar una imagen desde su dispositivo. Una vez cargada, la imagen seleccionada se muestra en la interfaz gráfica, reemplazando la imagen predeterminada. Entre las extensiones multimedia permitidas se encuentran JPG y PNG.

3.5.2. Configuración de opciones

La aplicación permite al usuario ajustar dos opciones antes de realizar la clasificación:

El usuario puede elegir el color de los *bounding boxes* entre una amplia gama; esto se debe a que ciertas imágenes pueden disimular los *bounding boxes* por los colores; especialmente si son pequeños.

El usuario también puede seleccionar lo que se denomina un *human threshold*. Cuando un modelo detecta un objeto, le da una probabilidad de pertenecer a una etiqueta, en este caso a “humano”. Es decir, el modelo generará un *bounding box* sobre un objeto si su porcentaje de pertenecer a la categoría “humano” es superior al umbral determinado por el usuario.

De esta forma podemos ajustar el modelo a nuestras necesidades. En ocasiones un usuario puede estar interesado en querer detectar todos los humanos de la imagen, independientemente de si otros elementos que no son humanos son detectados como tal. En ese caso un umbral bajo permitiría encontrar todos los humanos de la imagen. Cabe destacar que esta aproximación está muy relacionada con el concepto de un *recall*² alto y una *pre-*

²El *recall* (o sensibilidad) es una métrica utilizada en evaluación de modelos de clasificación, que indica la proporción de verdaderos positivos detectados con respecto al total de elementos relevantes. Es decir, mide la capacidad del modelo para encontrar todos los casos positivos. Explicada con detenimiento en posteriores capítulos

cisión³ baja. Ya que de los que realmente eran humanos, casi todos se van a detectar bien (*recall* alto) sin embargo, la precisión disminuirá porque de todos los detectados muchos no serán humanos (precisión baja).

En otras ocasiones podemos pretender que solo se detecten personas en el caso de que sea muy evidente que lo son, elevando el umbral. En este caso ocurre de manera contraria al anterior. La precisión será alta ya que solo se generan *bounding boxes* cuando se está muy seguro, y el *recall* será bajo ya que quedarán personas sin detectar por quedar debajo del umbral.

Aunque lo más pragmático es un equilibrio entre ambas aproximaciones, esto depende de lo que pretenda el usuario; es por eso que la aplicación proporciona dicha opción.

3.5.3. Proceso de clasificación

El modelo seleccionado procesa la imagen y genera *bounding boxes* alrededor de las siluetas que considera humanas.

El proceso se divide en varias etapas, cada una de las cuales se indica con mensajes de estado:

- **Carga del Detector:** Se carga el modelo seleccionado (*YOLOv4* o *YOLOX*) desde el archivo .mat correspondiente, Figuras 3.5 y 3.6.
- **Lectura de la Imagen:** La imagen cargada por el usuario se lee y se prepara para el procesamiento. Si es necesario, la imagen se redimensiona o normaliza para que sea compatible con el modelo.
- **Selección de opciones:** El usuario selecciona el color del *bounding box* y el *human threshold*, Figura 3.7
- **Detección de personas:** El modelo procesa la imagen y genera una lista de *bounding boxes*, junto con las probabilidades asociadas a cada detección. Solo se consideran las detecciones cuya probabilidad supera el umbral configurado por el usuario.
- **Generación de *Bounding Boxes*:** Figura 3.8 Se dibujan los cuadros delimitadores alrededor de las personas detectadas utilizando el color seleccionado por el usuario. Cada *bounding box* incluye la probabilidad de detección (*score*) como una etiqueta.

³La *precision* es una métrica utilizada en modelos de clasificación que indica la proporción de verdaderos positivos sobre el total de elementos que el modelo ha clasificado como positivos. Es decir, mide qué tan preciso es el modelo al identificar casos positivos. Vista con detenimiento en posteriores capítulos

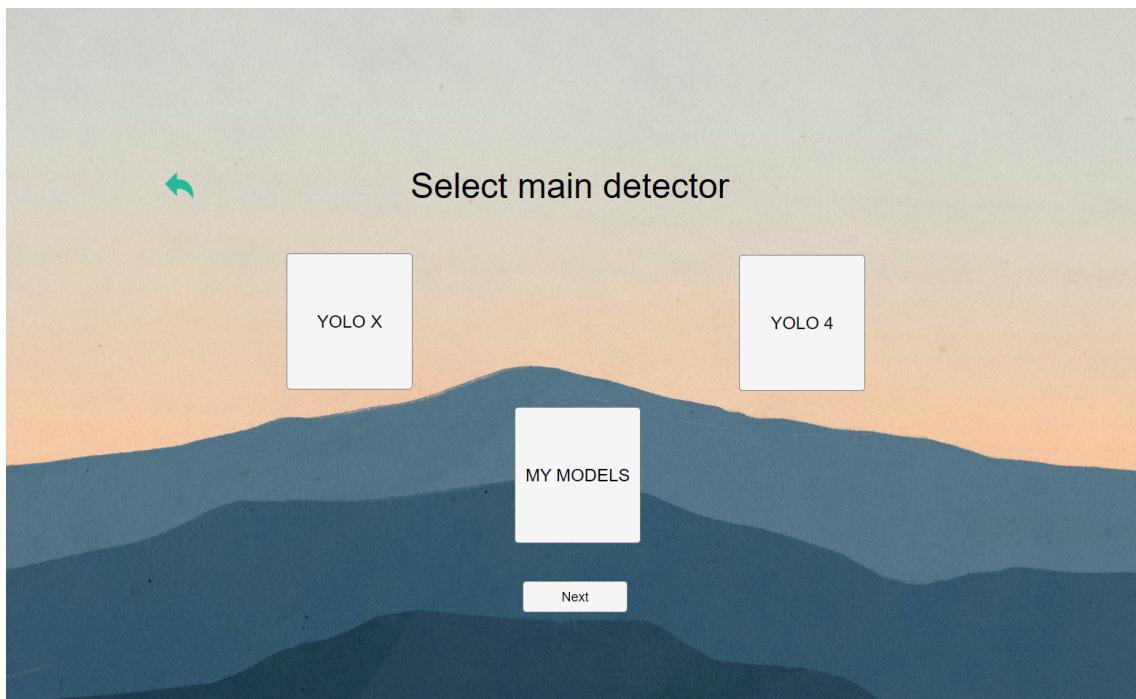


Figura 3.5: Selección de modelo para clasificar.

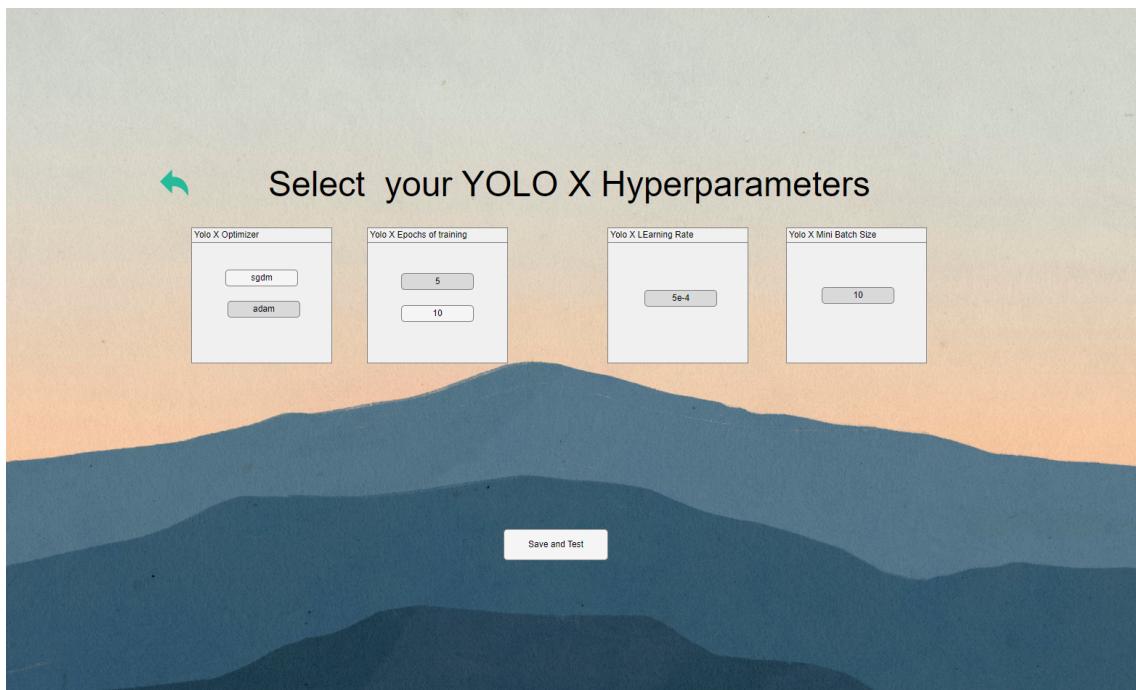


Figura 3.6: Selección de parámetros del modelo.

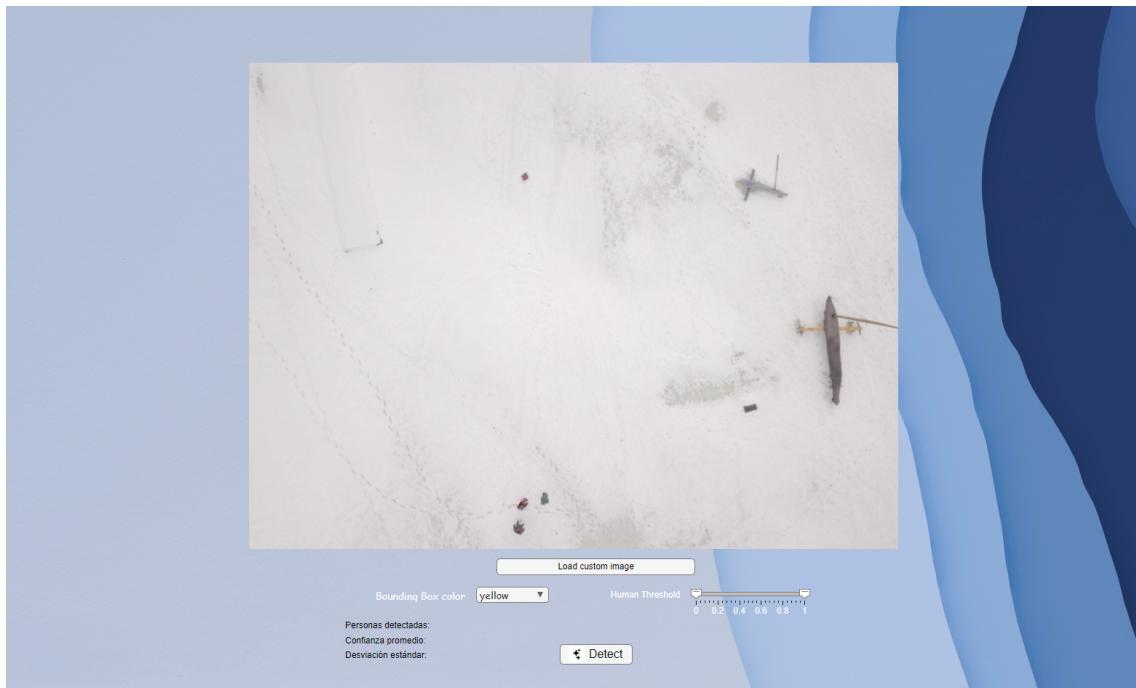


Figura 3.7: Selección de opciones para la clasificación.



Figura 3.8: Resultado final.

3.5.4. Visualización de resultados

Una vez completado el proceso de clasificación, la aplicación muestra los resultados de manera visual y estadística.

La imagen original mostrada en primer plano se actualiza para incluir los *bounding boxes*, las etiquetas de probabilidad y, además, los siguientes datos:

- Personas detectadas: Número total de personas detectadas en la imagen.
- Confianza promedio: Promedio de las probabilidades de las detecciones.
- Desviación estándar: Variabilidad en las confianzas de las detecciones. Un valor alto indica que algunas detecciones tienen confianzas muy diferentes al promedio.

3.5.5. Herramientas adicionales

- Navegación entre *bounding boxes*: El usuario puede navegar entre los *bounding boxes* generados para ver detalles ampliados.
- Descarga de la imagen clasificada: El usuario puede descargar la imagen con los *bounding boxes*.

3.6. Gestión del proyecto

3.6.1. Metodología

A la hora de llevar a cabo la gestión del proyecto se decidió emplear una metodología Scrumban. Esta elección se basa en el hecho de que los autores de este trabajo cuentan con experiencia en metodologías ágiles como Scrum. Se consideró que esta metodología presentaba un exceso de reuniones, las cuales no se amoldaban a las necesidades de un equipo tan pequeño de tres integrantes, por lo que podía resultar contraproducente y presentar dificultades organizativas.

Ante esta situación, se optó por incorporar aspectos de Kanban, dada su simplicidad y eficacia, sustentada en las siguientes normas fundamentales:

- **Visualizar el flujo de trabajo:** Permite conocer el estado de cada tarea en cualquier momento.

- **Determinar y respetar el WIP:** Se establecen límites al trabajo en curso para evitar la sobrecarga.
- **Gestionar el flujo del trabajo:** Se supervisa y optimiza la progresión de las tareas.
- **Establecer políticas explícitas:** Se definen reglas claras que facilitan la toma de decisiones.

Además, al no necesitar *sprints* para la realización de entregas periódicas, se consideró más adecuado implementar una metodología basada en un incremento continuo, lo que posibilitó una mayor flexibilidad en la gestión de las tareas.

Debido a estas consideraciones, se adoptó Scrumban como metodología a seguir. Manteniendo reuniones como las de sincronización, de retrospectiva tras la finalización de cada módulo o hito importante, sesiones de planificación para la incorporación de nuevas historias de usuario al tablero Kanban, y revisiones que debían ser realizadas por un integrante distinto al responsable del desarrollo de la historia, garantizando así una evaluación objetiva del trabajo realizado.

3.6.2. User Story Map e Historias de Usuario

Para organizar y priorizar las historias de usuario, se implementó un *User Story Map* en Miro. Esta herramienta permitió disponer las historias de usuario de forma visual, estructurándolas en diferentes niveles según su relevancia y prioridad dentro del flujo de desarrollo. De esta manera, resultaba más fácil visualizar el progreso del proyecto y cómo cada historia contribuía al objetivo general.

Cada historia de usuario seguía un formato estándar para mantener coherencia en su redacción:

“Como [tipo de usuario], quiero [funcionalidad o acción] para [beneficio o propósito]”.

Además, se aplicaron los criterios *SMART* e *INVEST* para asegurar que las historias fueran claras, útiles y bien definidas:

- **SMART** (*Specific, Measurable, Achievable, Relevant, Time-bound*): Específicas, Medibles, Alcanzables, Relevantes y con un Tiempo definido.
- **INVEST** (*Independent, Negotiable, Valuable, Estimable, Small, Testable*): Independientes, Negociables, Valiosas, Estimables, Pequeñas y Testeables.

Ambas son siglas provenientes del inglés y se utilizan comúnmente en metodologías ágiles para definir criterios de calidad en la redacción de historias de usuario.

En cuanto al nivel de detalle, se adoptó un enfoque intermedio. Si una historia de usuario era demasiado general o muy amplia, se desglosaba en historias más específicas, asegurando que fueran manejables y entendibles. Sin embargo, se evitó dividirlas en tareas, ya que esto implicaba un tiempo innecesario para el tamaño del equipo y no aportaba un beneficio real.

Cada historia era abordada en su totalidad por un miembro del equipo, lo que permitía que fluyera de manera más natural dentro del tablero *Kanban*. No obstante, cuando una historia resultaba especialmente compleja, se aplicaban estrategias colaborativas como el *swarming* (agrupamiento), en el que varios miembros trabajaban juntos en una misma historia. También se adoptó una política de “rey y sirvientes”, donde el miembro con la historia más prioritaria podía solicitar apoyo a alguien que estuviera trabajando en una tarea de menor prioridad.

3.6.3. Work in progress

El concepto de *WIP* (*Work in Progress*) es fundamental para la gestión ágil del trabajo, ya que se basa en limitar la cantidad de tareas en curso simultáneamente. Esta práctica sirve para evitar la dispersión del trabajo, reducir los tiempos de cambio entre actividades y garantizar que la calidad del trabajo se mantenga en un nivel óptimo.

En el proyecto, se implementó el límite *WIP* a tres tareas a la vez, asegurándose de que cada miembro del equipo se concentrara en finalizar lo que ya estaba en marcha antes de iniciar nuevas tareas. Para gestionar este flujo, se utilizó la aplicación Miro, que permitió diseñar un tablero digital visual y colaborativo.

Cada columna del tablero representaba una fase del proceso:

- **Backlog:** Historias de usuario identificadas, pendientes de ser priorizadas y trabajadas.
- **To do:** Historias seleccionadas para iniciar en el corto plazo.
- **En progreso:** Historias en desarrollo, organizadas visualmente según su grado de avance (las más a la izquierda eran recientes, mientras que las más a la derecha estaban avanzadas).
- **Haciendo pruebas:** Historias en fase de testeo para validar su correcto funcionamiento.
- **Revisión por otro miembro:** Para asegurar la calidad, las historias eran revisadas por un compañero distinto al desarrollador.

- **Terminada:** Historias completadas y validadas, asegurando que cumplieran con los estándares establecidos.

Este enfoque no solo mejoró la organización interna, sino que también ayudó a la comunicación y la toma de decisiones, ya que el equipo podía ver en tiempo real el estado de cada historia y realizar ajustes cuando fuera necesario.

Además, la implementación del *WIP* ayudó a mantener el equilibrio en la carga de trabajo del equipo. Si un miembro tenía demasiado trabajo en curso, se fomentaba la redistribución mediante estrategias como el *swarming* o la política de “rey y sirvientes”, asegurando que nadie quedara sobrecargado y que las historias más importantes avanzaran sin bloqueos.

En definitiva, la combinación del *WIP* y las estrategias colaborativas permitió optimizar la eficiencia y el rendimiento del equipo, asegurando un flujo de trabajo estable y efectivo.

3.6.4. Reuniones

Como ya se ha mencionado, las reuniones son imprescindibles para el correcto funcionamiento de las metodologías ágiles. En consecuencia, se ha determinado mantener las siguientes reuniones como parte esencial del proceso.

- **Sesiones de planificación:** Se organizaban para incorporar nuevas historias de usuario al tablero *Kanban*, definiendo prioridades y asignando tareas. En vez de hacer estas reuniones al comienzo del *sprint*, se realizaban una vez se terminaba un hito importante y se tenía que añadir más contenido para la entrega final.
- **Reuniones de retrospectiva:** Se convocaban al finalizar un módulo o al alcanzar un hito importante para evaluar lo realizado, identificar áreas de mejora y aprender de la experiencia.
- **Reuniones de sincronización:** Debido a la diferencia de horarios de los miembros del equipo, esta reunión solo se realizaba un par de días por semana, los lunes y los jueves. Estas reuniones eran cortas y en ellas se mostraba lo que se estaba realizando en ese momento y si existía alguna duda o problema con dicha tarea.
- **Revisiones:** Se estableció que las tareas fueran evaluadas por un compañero diferente al desarrollador y estando todos presentes por si alguno detectaba algún fallo o mal comportamiento, asegurando la calidad y la detección temprana de posibles errores.

3.6.5. Hitos

Los hitos del proyecto representan momentos clave dentro del desarrollo, donde se alcanzan objetivos parciales importantes. Estos puntos permiten evaluar el progreso, asegurar que se cumplen los plazos establecidos y detectar posibles retrasos o problemas en el plan original. En metodologías ágiles como *Scrumban*, los hitos ayudan a organizar el flujo de trabajo de manera más flexible, permitiendo realizar ajustes en función de los avances obtenidos. No todos los hitos están preestablecidos al comienzo, sino que van apareciendo a medida que el proyecto se desarrolla.

Dado que el objetivo del proyecto era la detección de personas en imágenes aéreas mediante algoritmos de aprendizaje profundo basados en modelos del tipo *CNN* y desarrollado en *MATLAB*, algunos de los hitos más relevantes fueron:

- Encontrar un *dataset* acorde con las necesidades del proyecto, es decir, imágenes de personas en el campo con anotaciones bien definidas y con una licencia que permitiera su uso para investigación.
- Aprender a usar *MATLAB* y entrenar el primer modelo de *YOLOv4* y posteriormente de *YOLOX*.
- Encontrar un servidor lo suficientemente potente para poder entrenar los modelos.
- Hacer el diseño de la *App*.
- Empezar el desarrollo de la *App* en *MATLAB*.

El seguimiento de estos hitos se realizó mediante el tablero *Kanban* en Miro, donde cada tarea relacionada con un hito se movía a través de las diferentes fases del tablero. Esto permitía visualizar el progreso de cada hito y detectar si alguno se demoraba más de lo previsto. En estos casos, se establecía una fecha límite y se priorizaba su finalización, asignando a todo el equipo para agilizar su finalización. Al finalizar cada hito importante, se realizaba una retrospectiva para evaluar el proceso y ajustar las siguientes tareas según las necesidades identificadas.

Los hitos fueron fundamentales para la organización del trabajo en el proyecto. En lugar de estructurar el desarrollo en *sprints* rígidos e inamovibles, se trabajó con un flujo continuo de tareas, incorporando nuevas historias de usuario tras la finalización de cada hito. Esta estrategia proporcionó una mayor flexibilidad y optimización de recursos, asegurando que el desarrollo avanzara de manera eficiente sin interrupciones. Además, evitó la presión innecesaria de cumplir plazos artificialmente ajustados, lo que permitió que cada tarea se completara con la calidad requerida antes de dar paso a la siguiente fase.

3.6.6. Gestión de configuración del software

La gestión de configuración del software es un proceso esencial en el desarrollo de aplicaciones, ya que permite controlar los cambios de las versiones en el código y documentos, ayudando a coordinar el trabajo en equipo. Para lograr una gestión más eficiente, se utilizaron diversas herramientas que facilitan la organización, comunicación y almacenamiento de archivos. Estas son las herramientas empleadas para el desarrollo del proyecto:

GitHub

Es una plataforma de control de versiones basada en *Git* que permite la gestión eficiente del código. Gracias a sus funcionalidades de gestión de ramas y cambios, es posible trabajar en equipo sin riesgo de sobrescribir el trabajo de otros. En el proyecto se usaron dos tipos de ramas: la *main*, donde se dejaba todo el código terminado y revisado por todos los integrantes del equipo, y las ramas *dev*, que se usaban para crear y desarrollar funcionalidades antes de integrarlas con el código principal, recibiendo nombres como: *dev-(HUE de la rama)*.

Google Drive

Se utilizó para el almacenamiento y compartición de documentos del proyecto, permitiendo que los miembros del equipo accedieran a la información en cualquier momento y desde cualquier dispositivo. Además, las herramientas de edición colaborativa de *Google Docs* facilitaron la elaboración de documentación.

Miro

Es una plataforma de colaboración visual empleada para la planificación y el diseño del proyecto, especialmente para el *User Story Map*, el tablero *Kanban* y algunos otros diagramas.

Gmail

Es un servicio de correo electrónico que permite enviar y recibir mensajes electrónicos de manera rápida y sencilla. Se utilizó para mantener contacto con los tutores.

Otras aplicaciones de mensajería

Para facilitar la coordinación y planificación de reuniones, se utilizaron aplicaciones de mensajería como *WhatsApp* y *Discord*. Estas plataformas permitieron una comunicación más ágil entre los miembros del equipo, el envío rápido de archivos y la organización de reuniones mediante grupos y canales dedicados al proyecto.

3.6.7. Riesgos

La gestión de riesgos es una parte fundamental en cualquier proyecto de software, ya que permite anticiparse a problemas que podrían afectar su éxito. Para ello, se adoptó una estrategia proactiva basada en el plan RSGR (Reconocimiento, Seguimiento, Gestión y

Resolución de riesgos), un método que permitió identificar, clasificar y priorizar los riesgos para minimizar su impacto.

A lo largo del desarrollo, se realizó un análisis detallado para detectar posibles riesgos en cada fase del proyecto. Estos se clasificaron según su probabilidad de hacerse realidad y su gravedad, lo que permitió diferenciar entre los riesgos más críticos y los de menor impacto. De esta manera, se gestionó de forma prioritaria el 20 % de los riesgos más serios, aquellos que podían comprometer el desarrollo del proyecto, asegurando así una respuesta eficaz y una mejor planificación.

Uno de los riesgos más relevantes fue la posibilidad de no contar con imágenes suficientes en el *dataset* para entrenar adecuadamente los modelos. Dado que el rendimiento de los algoritmos de detección depende directamente de la calidad y cantidad de datos, se consideró un riesgo de alta prioridad. Para mitigarlo, se estableció un criterio de búsqueda anticipada de bases de datos públicas con licencias adecuadas. En caso de no encontrar un volumen suficiente de imágenes, se contempló la posibilidad de capturarlas y anotarlas manualmente. Además, se evaluó la opción de generar datos sintéticos o aplicar técnicas de aumento de datos.

Otro riesgo significativo fue la necesidad de procesamiento en un servidor externo debido al alto coste computacional del entrenamiento con redes neuronales profundas. Este factor podía suponer tanto una barrera técnica como económica si los recursos locales eran insuficientes. Como plan de mitigación, se evaluaron varias opciones de entornos de cómputo en la nube, incluyendo plataformas académicas con acceso gratuito. El equipo también definió umbrales de rendimiento mínimo para los modelos, permitiendo priorizar entrenamientos eficientes durante las primeras fases del proyecto y escalar solo si era estrictamente necesario.

Para gestionar los riesgos, se siguieron tres enfoques clave: primero, se estableció un plan de prevención o mitigación para reducir la posibilidad de que ocurran; segundo, se implementó un plan de supervisión para evaluar si el riesgo se estaba volviendo más probable con el tiempo. Finalmente, se diseñó un plan de gestión o contingencia, que define las acciones a tomar en caso de que el riesgo se convierta en un problema real.

Capítulo 4

Análisis de resultados

En este capítulo se explican los principales recursos que han sido necesarios para la realización de la aplicación, además de las diferentes pruebas realizadas con los distintos modelos disponibles en la aplicación. Con respecto a la evaluación de los resultados se mostrarán varios ejemplos de cada modelo utilizado con sus respectivos análisis.

4.1. Recursos

En esta sección se explican brevemente el dataset utilizado junto con el hardware y el software que han sido necesarios.

4.1.1. Dataset y licencia

Una vez definido el tema de estudio y el tipo de elementos a identificar en las imágenes, se comenzó la búsqueda del dataset en plataformas que los ofrecen con una licencia para investigación y experimentación.

Se encontraron diversos datasets que podían ser útiles, pero finalmente se tomó la decisión de emplear Lacmus Drone Dataset (LADD)[18] debido a que las imágenes eran las más parecidas a lo que se tenía en mente y contaba con las anotaciones necesarias y una licencia apta para llevar a cabo la investigación.

Este dataset estaba inicialmente disponible en kaggle[19]. Sin embargo, posteriormente fue eliminado de dicha plataforma. Tras una búsqueda fue encontrado nuevamente, con

la misma licencia y el mismo conjunto de imágenes y datos en Dataset Ninja[20], otra plataforma que facilita la distribución de datasets especializada en computer vision para investigación y experimentación.

El Lacmus Drone Dataset (LADD)[18] es un conjunto de datos diseñado principalmente para tareas de búsqueda y rescate. Este dataset está distribuido bajo la licencia GNU GPL 3.0, lo que permite su uso libre para investigación y desarrollo, siempre y cuando se respeten los términos de dicha licencia, incluyendo la obligación de mantener la misma licencia en trabajos derivados y reconocer a los autores originales. Para más información sobre este tipo de licencia, consulte el archivo LICENSE, que se adjunta con el código.

4.1.2. Descripción del dataset

Este dataset incluye 1365 imágenes con un total de 4733 objetos etiquetados como "peatones". Dichos peatones están etiquetados utilizando cuadros delimitadores (*bounding boxes*) para facilitar el análisis y la detección.

Al haber 4733 personas en 1365 imágenes, hay una media de 3.47 personas por imagen, con un tamaño muy reducido y estar dispersas por toda la imagen, acompañado de que la cobertura promedio de las personas sobre la imagen es del 0.13 del área total, hace difícil su detección.

El tamaño y la resolución de las imágenes son variables. Esto se debe a que las imágenes fueron tomadas por drones operando en diferentes escenarios reales, lo que introduce diversidad en las proporciones y configuraciones visuales.

Las imágenes son muy variadas; encontramos entornos de todo tipo, como por ejemplo paisajes nevados (Figura 4.1), praderas (Figura 4.2), bosques frondosos (Figura 4.3) y llanuras (la Figura 4.4).



Figura 4.1: Ejemplo de entorno nevado



Figura 4.2: Ejemplo de pradera



Figura 4.3: Ejemplo de bosque frondoso



Figura 4.4: Ejemplo de llanura

4.1.3. *Hardware*

Durante las primeras etapas de desarrollo del proyecto, los recursos hardware de los que se disponían eran los equipos personales de los integrantes del proyecto. Según se fueron realizando los primeros entrenamientos se pudo observar cómo los modelos *YOLO*, de por si pesados, unidos a la gran cantidad de imágenes que procesar y al gran tamaño de las mismas, hacía que se necesitasen sesiones de cómputo de más de un día para poder obtener algún modelo entrenado con algunas pocas etapas, además de dejar inutilizados los equipos durante los entrenamientos, y teniendo que hacer pausas durante las noches.

Debido a que este problema dificultaba la correcta y coherente realización del trabajo se decidió utilizar uno de los servidores disponibles en el grupo de investigación ISCAR (Ingeniería de Sistemas, Control, Automatización y Robótica) de la Facultad de Informática, con alta capacidad de cómputo, tanto en procesamiento como en memoria. De esta forma las pruebas se desarrollaron en este servidor en el que se habilitó una cuenta específica para los miembros del proyecto.

Tras unas primeras pruebas se pudo observar cómo dicho servidor contaba con gran cantidad de recursos, como mayor cantidad de memoria *RAM* y tarjeta gráfica, que hacían mucho más corto el proceso de entrenamiento. Los estudiantes podían conectarse al servidor desde sus equipos personales y programar entrenamientos en horas específicas del día por turnos. Por consiguiente se logró en poco tiempo llevar a cabo el entrenamiento de diversos modelos con una gran diversidad de parámetros, lo que permitió a los alumnos hacer modificaciones e ir perfeccionando y logrando mejores resultados.

4.1.4. *Software*

El servidor proporcionado por el grupo de investigación ISCAR cuenta con un sistema operativo Ubuntu 20.04.6 LTS; desde ese entorno fueron empleadas dos versiones de MATLAB, ambas lanzadas en 2024, la R2024a y R2024b, siendo esta última la empleada hasta la finalización del proyecto. Dentro del propio MATLAB se usaron varios *Toolboxes* específicos para facilitar el desarrollo del proyecto, estos son:

- ***Computer Vision Toolbox***: Proporciona un conjunto de algoritmos y funciones para diseñar y evaluar sistemas de visión artificial.
- ***Automated Visual Inspection Library for Computer Vision Toolbox***: Extiende las capacidades del *Computer Vision Toolbox* e incluye el *YOLOX* usado en el proyecto.
- ***Computer Vision Toolbox Model for YOLOv4 Object Detection***: Implementa el modelo *YOLOv4* para detección de objetos.
- ***Curve Fitting Toolbox***: Añade funciones para ajustar curvas y superficies a datos,

permitiendo realizar análisis exploratorio de datos, procesar datos, comparar modelos candidatos entre otros.

- **Deep Learning Toolbox:** Proporciona funcionalidades y herramientas para implementar modelos de redes de *Deep Learning*.
- **Deep Learning HDL Toolbox:** Facilita la implementación y optimización de redes de *Deep Learning* en hardware *FPGA* y *ASIC*. Es una extensión de *Deep Learning Toolbox*.
- **Image Processing Toolbox:** Ofrece funciones para procesar, analizar y visualizar imágenes, añadiendo operaciones de filtrado, segmentación, transformaciones entre otras muchas.
- **Parallel Computing Toolbox:** Optimiza el rendimiento de problemas que requieren gran cantidad de datos y alta carga computacional, mediante el uso de procesadores multinúcleo, *GPU* y *clústeres* de procesamiento.
- **Statistics and Machine Learning Toolbox:** Proporciona métodos estadísticos y de aprendizaje automático para la modelación y el análisis de datos, incluyendo herramientas para regresión, clasificación, agrupamiento y análisis exploratorio.

Además de estos *Toolboxes* también se probó un pequeño programa en *python* con la librería *kmeans* de *scikit learn*, para reducir la cantidad de colores de las imágenes para simplificarlas y ver si ayudaba a mejorar los resultados del aprendizaje.

Finalmente, en cuanto a *software*, también se usaron algunos *scripts* en *bash* para programar las ejecuciones de entrenamiento y evitar solapamientos entre dos ejecuciones, comunicándose con un *bot* de *telegram* para conocer cuándo empieza y acaba y los resultados de los entrenamientos, o si saltó un error en medio de la ejecución.

4.2. Análisis de resultados

En este punto se procede a la exposición de los pasos seguidos por el equipo de desarrollo para la obtención de resultados, así como a la presentación de los mismos. De este trabajo se derivan diferentes tipos de resultados, algunos, como la aplicación diseñada, ya han sido abordados ampliamente en el capítulo 3. Es por ello que, en este punto, se detallan los resultados pertenecientes a la actividad de investigación sobre el comportamiento de los modelos en el *Lacmus Drone Dataset* (LADD) tanto desde el punto de vista del entrenamiento como de la clasificación.

4.2.1. Obtención de resultados

Antes de centrarse en los resultados, se procede a la explicación de ciertos conceptos que serán de utilidad para su correcta comprensión e interpretación. Además, se presentan las pautas seguidas en los entrenamientos que han permitido obtener dichos resultados.

Es fundamental considerar que la obtención de buenos resultados pasa, en primer lugar, por llevar a cabo un proceso de aprendizaje que dote al modelo de robustez y precisión ante imágenes que no ha visto previamente, es decir, una buena capacidad de generalización. Con este objetivo, el equipo de desarrollo consideró que la mejor manera de emplear el conjunto de datos era mediante la técnica *Train-Test-Validation Split* (Figura 4.5). Esta técnica permite la división del *dataset* en tres conjuntos de datos principales.

El primero de ellos es el conjunto de entrenamiento, que requiere un mayor volumen de datos, en este caso, imágenes. Cuanta mayor sea la diversidad de ejemplos, mejor será la generalización del modelo. En este caso, se determinó que el 75 % de las imágenes disponibles se destinaran a entrenamiento.

A continuación, se encuentra el conjunto de validación, cuya función es comprobar el rendimiento de los modelos tras el entrenamiento, ya que contiene imágenes nunca antes vistas por el modelo. Puede parecer similar al conjunto de test; sin embargo, la diferencia principal es que, gracias a los resultados obtenidos con el conjunto de validación, es posible determinar cuáles son los hiperparámetros (Figura 4.5) que mejores resultados ofrecen. Una vez establecida la configuración óptima, se realiza una última validación empleando el conjunto de test. De esta forma, se obtiene una visión más clara del comportamiento de los modelos y se identifican las mejores variantes, lo que no sería tan sencillo si se eliminara el conjunto de validación y solo se emplearan entrenamiento y test.

Para este caso, el conjunto de validación representa un 10 % del dataset, mientras que el conjunto de test constituye el 15 % restante.

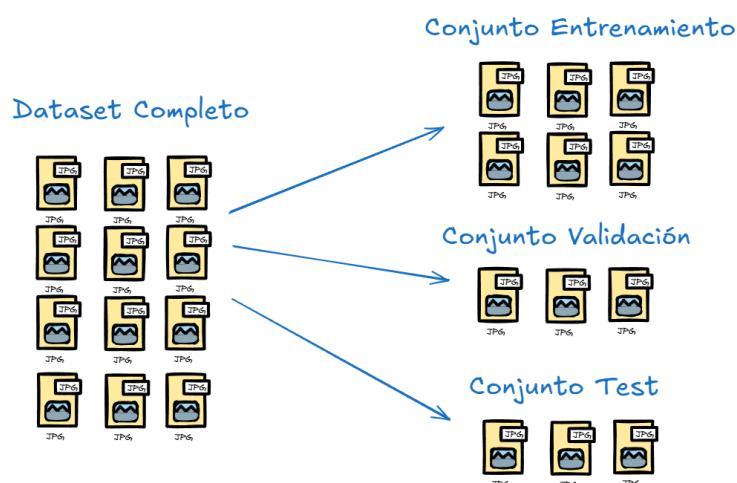


Figura 4.5: Visualización división del dataset, Fuente: Propia

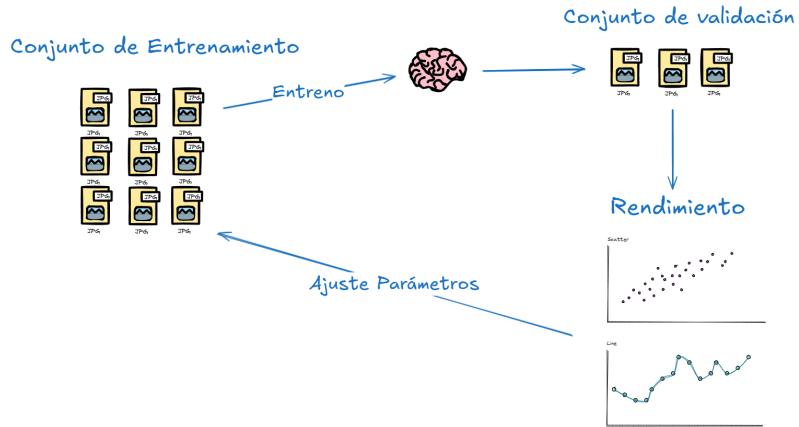


Figura 4.6: Proceso de entrenamiento y validación, Fuente: Propia

4.2.2. Métricas

Para poder evaluar correctamente el rendimiento de un modelo de detección de objetos, como *YOLOv4* o *YOLOX*, es necesario usar una serie de métricas. Estas métricas permiten comparar diferentes modelos o configuraciones, así como identificar puntos fuertes y débiles en su capacidad de detección. En el caso específico de la detección de objetos en imágenes, estas métricas toman en cuenta tanto la precisión en la localización (*bounding boxes*) como la clasificación correcta de los objetos.

A continuación se describen las métricas más utilizadas en este tipo de tareas:

1. **Precisión (Precision)** La precisión (P) mide la proporción de verdaderos positivos sobre el total de elementos detectados como positivos por el modelo. Es decir, de todas las detecciones realizadas, cuántas son realmente correctas. Se calcula con la siguiente fórmula:

$$Precision = \frac{VP}{VP + FP} \quad (4.1)$$

donde VP son los verdaderos positivos (detecciones correctas) y FP los falsos positivos (detecciones incorrectas). Una alta precisión indica que el modelo comete pocos errores a la hora de identificar objetos.

2. **Recall** El recall (R) mide la capacidad del modelo para detectar todos los objetos relevantes presentes en una imagen. Se expresa como:

$$Recall = \frac{VP}{VP + FN} \quad (4.2)$$

donde FN son los falsos negativos, es decir, objetos que estaban en la imagen pero que el modelo no detectó. Un valor alto de *recall* implica que el modelo rara vez deja sin detectar un objeto que debería identificar.

3. **Accuracy** La *accuracy* o exactitud mide el porcentaje de predicciones correctas (tanto verdaderos positivos como verdaderos negativos) sobre el total de casos evaluados:

$$\text{Accuracy} = \frac{VP + VN}{VP + VN + FP + FN} \quad (4.3)$$

Aunque es una métrica común en clasificación, en tareas de detección de objetos suele tener menor relevancia, debido a que la cantidad de verdaderos negativos (VN), áreas sin objetos suele ser muy alta por lo que suele distorsionar la interpretación del resultado.

4. **F1-Score** El *F1-score* es una métrica que combina la precisión y el *recall*, es realmente útil cuando se busca un equilibrio entre ambas. Se trata de la media armónica entre precisión y *recall*:

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4.4)$$

Un *F1-score* alto indica que el modelo tiene tanto una buena precisión como un buen *recall*.

5. **Curva Precision-Recall (PR Curve)** Para obtener una visión más completa del rendimiento del modelo, se puede representar gráficamente la curva de precisión vs. *recall*. Esta curva se construye calculando distintos valores de precisión y *recall* para diferentes umbrales de confianza (*confidence threshold*). El *recall* se representa en el eje X y la precisión en el eje Y, como se puede ver en la Figura 4.7. Esta visualización permite entender cómo varía el rendimiento del modelo al modificar el criterio de decisión para aceptar una predicción como válida.

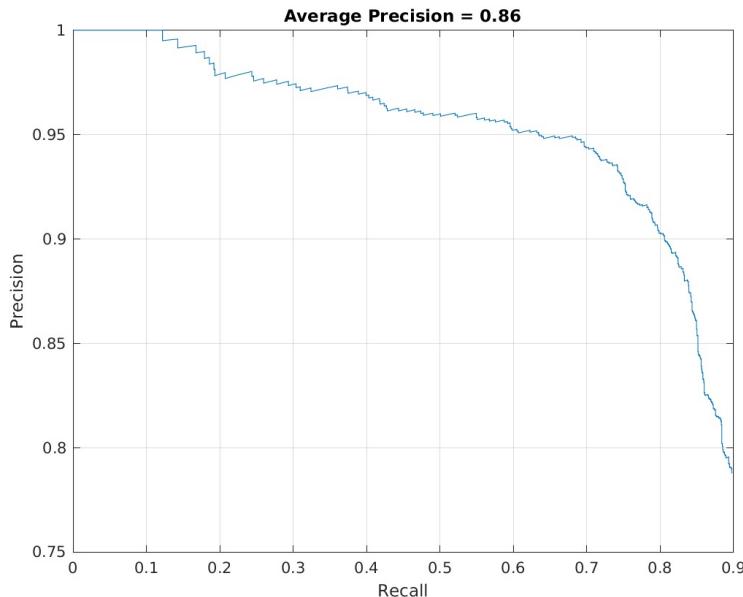


Figura 4.7: Ejemplo de la Curva *Precision Recall*

6. **Average Precision (AP)** El *Average Precision*, AP (precisión media) es una métrica que resume la curva PR en un solo número. Siendo el área bajo la curva PR.

Para calcularla, se integra sobre los valores de R en el eje de abscisas entre 0 y 1. Como el cálculo del área bajo una curva es computacionalmente costoso, lo que se hace en la práctica es una interpolación tal y como se define a continuación

$$AP = \int_0^1 precision(r) dr \Rightarrow AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i) \cdot p_{\text{int}}(r_{i+1}) \quad (4.5)$$

7. ***mAP50*** Una métrica muy común en *benchmarks* de detección es el mAP (*mean Average Precision*). Esta métrica calcula la media de las APs obtenidas para cada clase del conjunto de datos. En el caso de que solo se esté detectando una clase (por ejemplo, personas), el mAP coincide con el AP de dicha clase.

El índice 50 indica el umbral de IoU (*Intersection over Union*) utilizado para considerar una detección como correcta. En concreto, mAP50 requiere que la intersección entre el bounding box predicho y el objeto real (*ground truth*) sea al menos del 50 %. Esta métrica proporciona una buena idea general de la precisión espacial de las predicciones.

4.2.3. Análisis de resultados

A continuación se muestran los resultados más relevantes obtenidos con los modelos *YOLOv4* y *YOLOX*.

4.2.3.1. *YOLOv4*

- **Modelos entrenados (*overview*)**

A continuación se muestran distintas opciones de entrenamiento, indicando en cada caso, cuál es el mejor.

En el caso de *YOLOv4*, se ha llevado a cabo un exhaustivo proceso de experimentación mediante el entrenamiento de múltiples variantes del modelo, ajustando diferentes combinaciones de hiperparámetros. Las configuraciones consideradas incluyen dos tipos de optimizadores (ADAM y SGDM), distintos valores de *epochs*, valores de *learning rate* y tamaños de *minibatch*.

A continuación, se detallan los modelos entrenados:

- **Optimizador ADAM:**

- 25 *epochs*, *learning rate* = 0.0015, *minibatch* = 6 y 8
- 25 *epochs*, *learning rate* = 0.001, *minibatch* = 6 y 8
- 50 *epochs*, *learning rate* = 0.0015, *minibatch* = 6 y 8
- 50 *epochs*, *learning rate* = 0.001, *minibatch* = 6 y 8

- **Optimizador SGDM:**

- 25 *epochs*, *learning rate* = 0.0001, *minibatch* = 16 y 8

- 25 epochs, learning rate = 0.0002, minibatch = 16 y 8
- 50 epochs, learning rate = 0.0001, minibatch = 16 y 8
- 50 epochs, learning rate = 0.0002, minibatch = 16 y 8
- 75 epochs, learning rate = 0.0001, minibatch = 16 y 8
- 75 epochs, learning rate = 0.0002, minibatch = 16 y 8

■ Entrenamiento

Durante el proceso de entrenamiento, el equipo de desarrollo pudo ir monitorizando cómo se comportaba el modelo. según se iba llevando a cabo el entrenamiento. La información analizada durante este proceso es la evolución de la pérdida en el conjunto de entrenamiento *Training loss* y la pérdida en el conjunto de validación *Validation loss*. A continuación, se muestra la figura generada durante el proceso de entrenamiento.

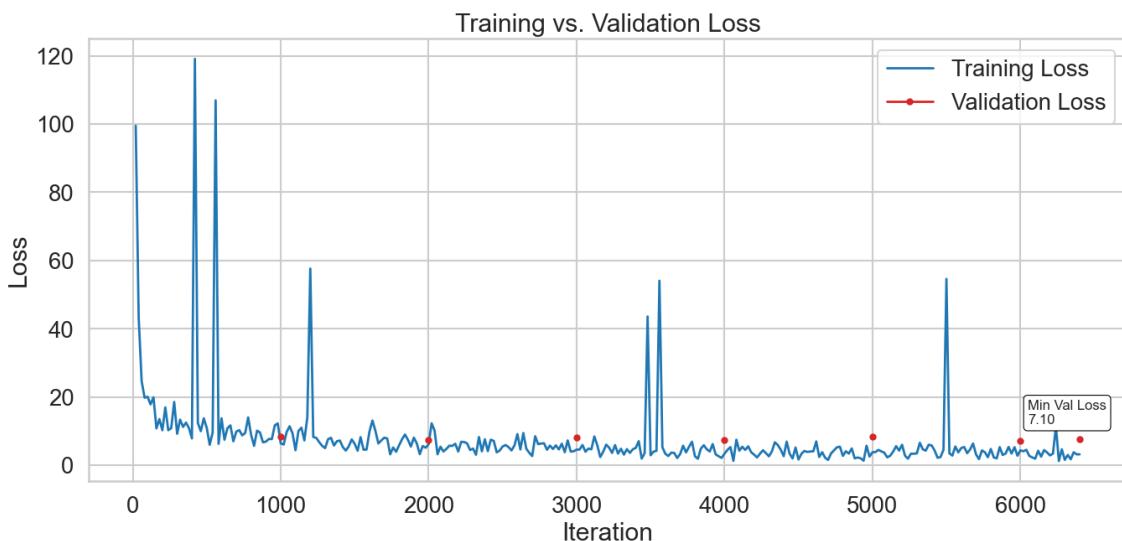


Figura 4.8: Ejemplo de evaluación de métricas en entrenamiento de un modelo YOLOv4

Uno de los factores a tener en cuenta es que estos modelos requieren una gran cantidad de horas de cómputo para completar el proceso de entrenamiento. Por ejemplo, realizar un entrenamiento de tan solo 10 épocas puede suponer aproximadamente una hora y media de cómputo, sin que ello garantice necesariamente obtener buenos resultados.

El entrenamiento más prolongado llevado a cabo consistió en 100 épocas, superando las 13 horas de procesamiento. No obstante, conviene destacar que una mayor duración del entrenamiento no implica automáticamente una mejora en el rendimiento del modelo. Tal y como se explicó anteriormente, el sobreajuste (*overfitting*) es uno de los principales problemas cuando se sobreentrena un modelo, y su detección temprana se convierte en una herramienta fundamental para evitarlo.

Como se puede apreciar en la Figura 4.9, el valor mínimo de pérdida en el conjunto de validación no se alcanza al completar las 100 épocas de entrenamiento, como podría esperarse. De hecho, el mejor resultado (6.64) se obtiene antes de la época

50, lo que indica que un entrenamiento más prolongado no siempre conduce a mejores resultados.

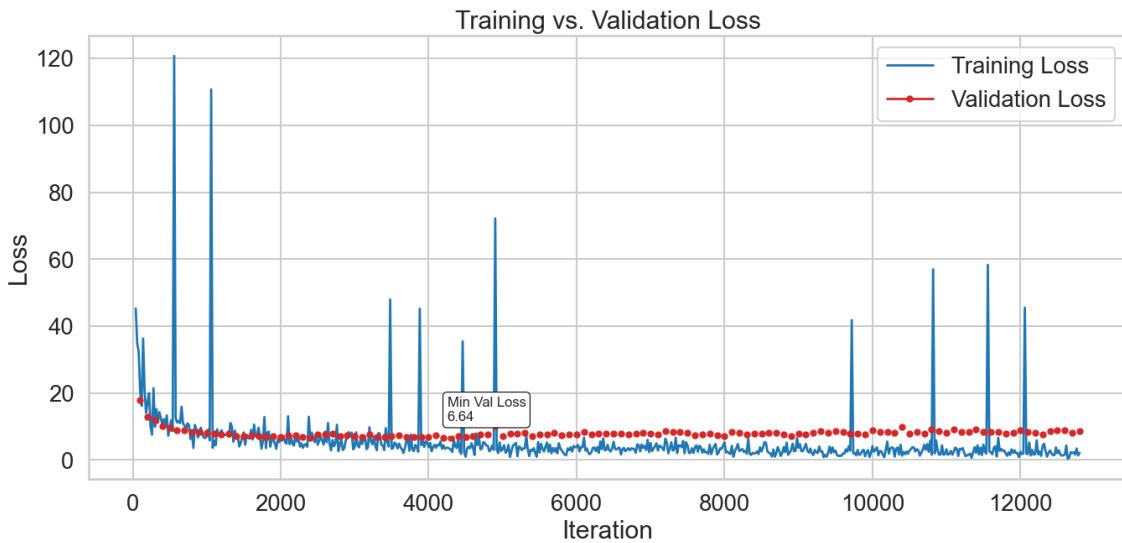


Figura 4.9: Ejemplo de evaluación de métricas en entrenamiento de un modelo YOLOv4 de 100 épocas

■ Test

Para determinar qué modelo era capaz de generar los mejores resultados, se decidió emplear como métricas la precisión y el *recall* obtenidos por parte de cada modelo frente al conjunto de test. Sin embargo, los resultados obtenidos por todas las variantes de *YOLOv4* (combinaciones de hiperparámetros para el modelo *YOLOv4*) no fueron los esperados, ya que todos los modelos no eran capaces de generalizar bien y obtener buenos resultados frente a imágenes nunca vistas por el modelo. La variante que mejores resultados obtuvo en esta prueba de test fue el modelo con la siguiente configuración de hiperparámetros: ADAM, 50 epochs, learning rate = 0.001 y minibatch = 8 el cual presentaba una precisión de 0.14.

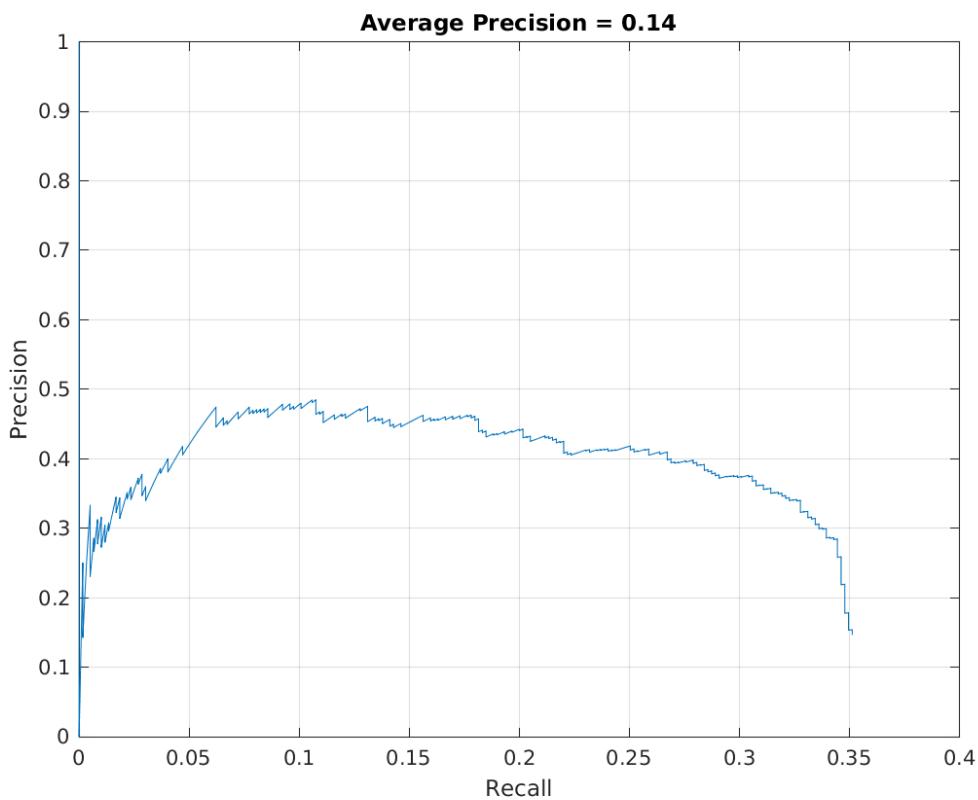


Figura 4.10: Resultado obtenido por el mejor modelo YOLOv4 para el conjunto de test

A partir de la gráfica mostrada en la Figura 4.10 se puede interpretar lo siguiente: a medida que se incrementa el *recall*, la precisión del modelo disminuye. Esto implica que, para lograr que el sistema detecte a la mayoría o la totalidad de las personas presentes en las imágenes, es necesario aceptar una menor precisión. En otras palabras, el modelo genera un mayor número de cajas delimitadoras (*bounding boxes*), pero muchas de ellas son incorrectas o no corresponden realmente a una persona. Es decir, se aumenta la capacidad del sistema para no dejar personas sin detectar a costa de una mayor cantidad de falsos positivos.

4.2.3.2. *YOLOX*

- **Modelos entrenados (*overview*)**

En el caso de *YOLOX* se decidió explorar un segundo modelo con el fin de comparar y determinar si modelos de la misma familia de *YOLO* pero posteriores en el tiempo presentaban mayor robustez frente al *dataset* elegido. El objetivo era demostrar una evolución considerable frente a imágenes de mayor complejidad debido al tamaño reducido de los objetos presentes en ellas.

A continuación, se detallan los parámetros utilizados para el entrenamiento:

- Optimizador Adam:

- 5 epochs, learning rate = 0.0005, minibatch = 5
- 5 epochs, learning rate = 0.0005, minibatch = 10
- 10 epochs, learning rate = 0.0005, minibatch = 10

- Optimizador SGDM:

- 5 epochs, learning rate = 0.0005, minibatch = 5
- 5 epochs, learning rate = 0.0005, minibatch = 10
- 10 epochs, learning rate = 0.0005, minibatch = 10

- Entrenamiento

Durante el proceso de entrenamiento, al igual que ocurrió con *YOLOv4*, se pudo analizar el comportamiento de los modelos en tiempo real. Tanto la pérdida en entrenamiento como en validación eran visibles, lo que permitió observar que, con muchas menos épocas, los modelos basados en *YOLOX* conseguían reducir significativamente las pérdidas, obteniendo mejores resultados en un tiempo considerablemente menor.

Además, la métrica AP50, explicada previamente, arrojaba valores prometedores. En este punto, el equipo de desarrollo tomó conciencia de que esta nueva aproximación representaba una mejora sustancial en la resolución del problema, a falta de confirmar esta hipótesis con los resultados obtenidos sobre el conjunto de test. La Figura 4.11 muestra la evolución de las métricas del proceso de entrenamiento del modelo que emplea ADAM como optimizador, 5 epochs, learning rate = 5×10^{-4} , minibatch = 5.

Obsérvese cómo a medida que el número de iteraciones aumenta, la validación experimenta un incremento que tiende al 100 % de eficacia, de forma incremental, mientras la pérdida evoluciona aproximándose al valor cero, que es el ideal.

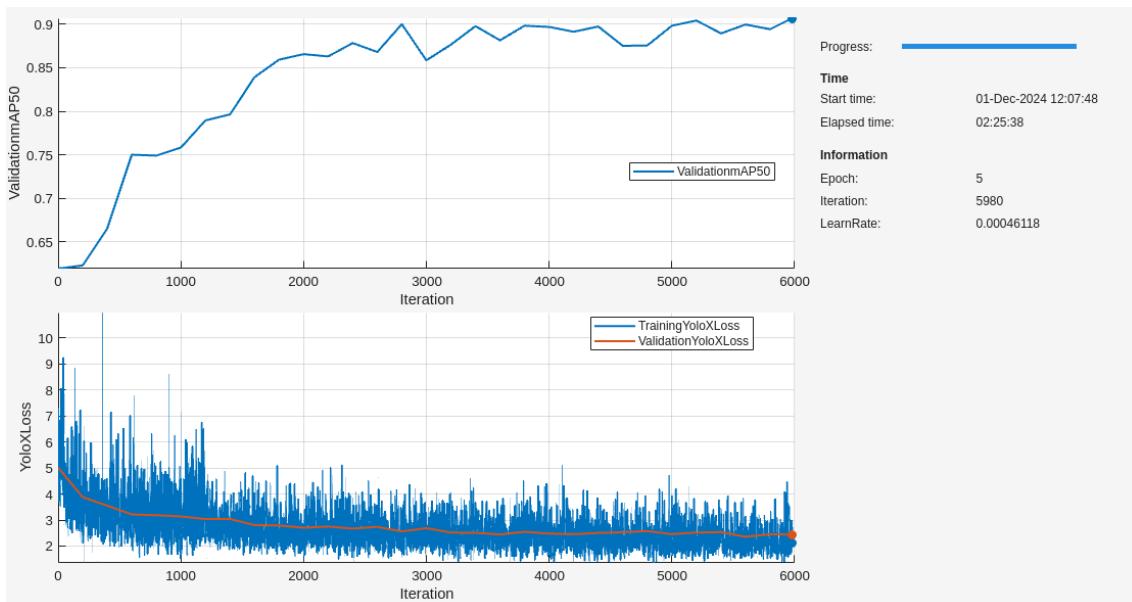


Figura 4.11: Ejemplo de evaluación de métricas en el proceso de entrenamiento de un modelo *YOLOX*

- **Test**

De manera análoga a los modelos de *YOLOv4* se decidió emplear la gráfica de precisión y *recall* como método de evaluación sobre el conjunto de test. De esta forma se pudo observar que todos los modelos entrenados presentaban comportamientos muy similares y positivos. Se corroboró la hipótesis que se había arrojado durante la fase de entrenamiento; los modelos de *YOLOX* son capaces de obtener mejores resultados a la hora de detectar las personas en el dataset seleccionado. No solo los resultados en imágenes nunca vistas son buenos, sino que los tiempos de cómputo requeridos son mucho menores. La gráfica 4.12 muestra la relación entre la precisión y el *recall* del mejor de los modelos, que es el siguiente: *ADAM*, 10 *epochs*, *learning rate* = 5×10^{-4} , *minibatch* = 10. Obteniendo un 0.86 de precisión, que contrasta con el 0.14 obtenido por el mejor de los modelos de *YOLOv4*.

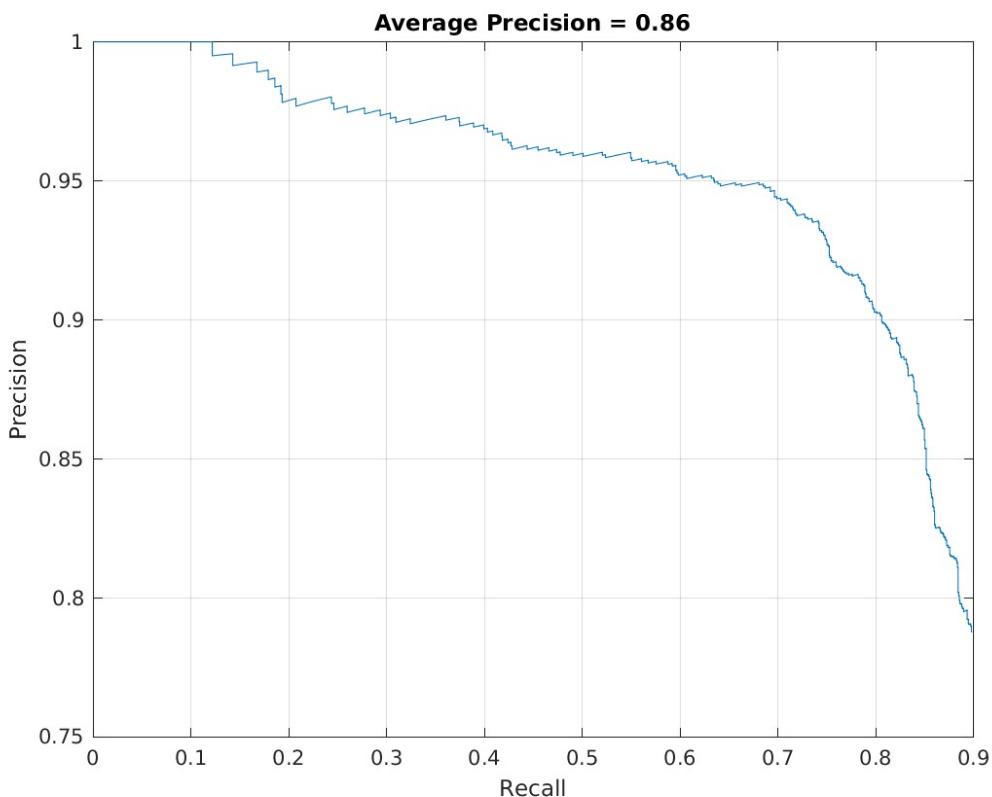


Figura 4.12: Resultado obtenido por el mejor modelo *YOLOX* para el conjunto de test

En este caso se observa cómo a medida que el *recall* aumenta la precisión disminuye, pero sigue manteniendo valores muy altos, por lo que esto da a entender que para que el sistema no deje personas sin clasificar no es necesario sacrificar falsos positivos, ya que la precisión sigue siendo alta, aportando un equilibrio que permite concluir que el uso de estos modelos es preferente frente a los modelos de *YOLOv4*.

4.2.3.3. Comparativa en ejemplos reales

Algunos ejemplos de imágenes etiquetadas por modelos *YOLOv4* y *YOLOX*

Cabe mencionar que los siguientes ejemplos han sido generados empleando los dos modelos que mejores resultados han obtenido en la fase mencionada con anterioridad.



Imagen 667 etiquetada por *YOLOv4*



Imagen 667 etiquetada por *YOLOX*

Figura 4.13: Comparación de la imagen 667 etiquetada por *YOLOv4* y *YOLOX*

La imagen 667 del *dataset* contiene ocho personas. Tras el proceso de clasificación utilizando *YOLOv4*, se obtiene un resultado alejado del esperado: del primer grupo de personas ubicado en la parte inferior de la imagen, solo una es detectada; de forma análoga, en el segundo grupo, situado en la parte superior, ocurre lo mismo. Además, se genera una *bounding box* en una zona "vacía", donde no hay ninguna persona.

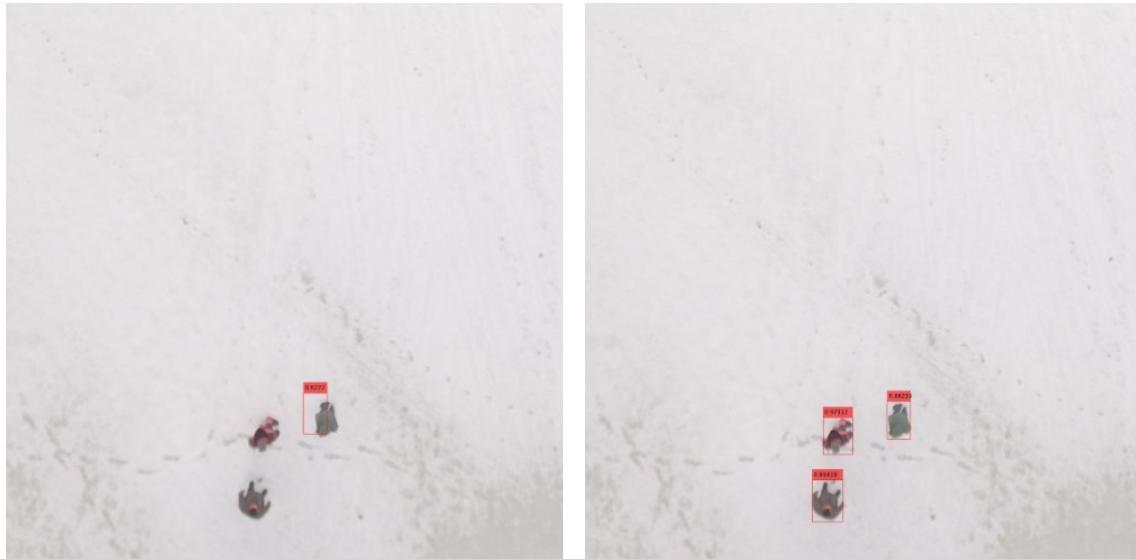
En contraste, el resultado obtenido mediante *YOLOX* es considerablemente mejor: el grupo de la parte inferior es reconocido correctamente y, en el segundo grupo, no se generan *bounding boxes* en zonas vacías.

Imagen 1259 etiquetada por *YOLOv4*Imagen 1259 etiquetada por *YOLOX*Figura 4.14: Comparación de la imagen 1259 etiquetada por *YOLOv4* y *YOLOX*

En la figura de la derecha se observa el correcto desempeño del modelo de *YOLOX* frente a la imagen 1259, en la que se encuentran 3 personas y todas son detectadas por el modelo. Sin embargo, *YOLOv4* tan solo es capaz de detectar una de ellas.

Imagen 824 etiquetada por *YOLOv4*Imagen 824 etiquetada por *YOLOX*Figura 4.15: Comparación de la imagen 824 etiquetada por *YOLOv4* y *YOLOX*

En este ejemplo se demuestran las capacidades de *YOLOX* para ser capaz de detectar con mucha precisión las siluetas humanas. Mientras que *YOLOv4* detecta ambas personas como una sola, *YOLOX* es capaz de diferenciar entre ambas y generar dos *bounding boxes*.

Imagen 1259 etiquetada por *YOLOv4*Imagen 1259 etiquetada por *YOLOX*Figura 4.16: Comparación de la imagen 1259 en condiciones de nieve, etiquetada por *YOLOv4* y *YOLOX*

En este último ejemplo se muestra la tendencia mencionada con anterioridad, pero en una imagen en un entorno diferente, en este caso en entorno nevado. Se observa cómo se mantienen los mismos problemas que en los casos anteriores, *YOLOv4* no es capaz de detectar a las 3 personas mientras que *YOLOX* genera los *bounding boxes* correctamente.

Como conclusión, se puede mencionar que *YOLOv4*, independientemente de la configuración de parámetros, no es la mejor aproximación para la resolución de un problema de detección de objetos cuando estos son de tamaño reducido, ya que en diversas ocasiones los *bounding boxes* no se generan correctamente o no se generan en los lugares donde deberían. Por su lado *YOLOX* presenta una mayor robustez para resolver este problema, siendo capaz de delimitar correctamente las siluetas humanas por pequeñas que sean y por unidas que estén a otras.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

Hace un año, cuando se propuso la idea de este TFG, entre los integrantes del grupo se generó un alto nivel de entusiasmo. Los tres miembros presentaban una gran curiosidad hacia la Inteligencia Artificial, gracias a la realización de algunas asignaturas durante el grado, muchas de ellas centradas en el aprendizaje automático. Sin embargo, las técnicas estudiadas trabajaban, en su mayoría, sobre datos tabulares. Es por ello que la realización de un proyecto de tal envergadura —un TFG sobre la detección de personas en imágenes— suponía una gran oportunidad para conocer una variante del aprendizaje automático que, hasta ese momento, se desconocía.

Tras la realización del TFG, los tres autores coinciden en que ha supuesto una gran experiencia y un proceso de aprendizaje muy completo. Llevar a cabo este trabajo ha permitido a los autores ampliar su conocimiento teórico acerca de los modelos y técnicas subyacentes. Conceptos que antes se desconocían, como las redes neuronales convolucionales o las operaciones de agrupamiento, ya no resultan extraños y, desde este momento, se incluyen en los conocimientos adquiridos tras la realización de este proyecto. Por otro lado, la realización de este trabajo ha permitido el acercamiento a una nueva tecnología nunca antes estudiada, como es *Matlab* y su herramienta de diseño de aplicaciones, *Matlab App Designer*. Tras su uso, se considera que esta opción es tan válida para llevar a cabo estas tareas como otras, en un principio más llamativas, como pudiera ser *Python* y sus librerías asociadas. El equipo de desarrollo se ha sentido cómodo desde el primer momento con el entorno y, gracias a una clara y abundante documentación, no se han encontrado inconvenientes de ninguna índole.

Cabe mencionar que, durante este periodo, no solo se han adquirido fundamentos teóricos. Un TFG, debido a su complejidad, requiere de un alto nivel de cohesión, estructura y planificación por parte de los integrantes. Todos coinciden en que una buena organización es fundamental para la correcta realización de cualquier actividad de este estilo, y que, en

algunos casos, es tan importante como los conceptos teóricos que se pretenden explorar. Es por ello que, durante estos meses, se ha insistido en seguir pautas para evitar el descontrol o sustos de última hora.

Además, este trabajo ha sido una excelente oportunidad para que los componentes del equipo se enfrenten de manera real a los desafíos y dinámicas de un proyecto de investigación aplicada. Desde establecer objetivos hasta evaluar resultados, han tenido que lidiar con problemas típicos del desarrollo de sistemas de visión por computador, como el tratamiento de imágenes, la gestión de grandes conjuntos de datos y la evaluación del rendimiento de los modelos entrenados. En lugar de ser un obstáculo insuperable, estas dificultades han sido una fuente constante de aprendizaje, fortaleciendo habilidades como la resiliencia, el pensamiento crítico y la capacidad de análisis.

Además, se ha promovido un enfoque autodidacta para profundizar en aspectos técnicos que no se cubren directamente en el grado, como la comprensión de arquitecturas de redes neuronales avanzadas (como *YOLO* y *Faster R-CNN*) o técnicas para mejorar el rendimiento mediante regularización, aumento de datos o ajustes de hiperparámetros. Todo esto ha despertado un mayor interés en la investigación en inteligencia artificial, abriendo la puerta a posibles estudios de máster o incluso a futuras oportunidades laborales en este campo.

Por último, es crucial resaltar el impacto personal que este TFG ha tenido en los integrantes del grupo. Más allá del conocimiento técnico adquirido, el proyecto ha servido como una validación de las propias capacidades y ha fomentado un sentido de logro que va más allá de lo académico.

Por todo lo expuesto, el equipo queda muy satisfecho con el trabajo realizado y agradecido por la oportunidad que les ha sido otorgada por parte de su tutor para llevarlo a cabo.

Como conclusión a lo mencionado, conviene destacar que se han cumplido satisfactoriamente todos los objetivos que fueron propuestos al inicio del proyecto. Estos pueden apreciarse a través de la aplicación desarrollada, una aplicación de detección de objetos empleando técnicas de *Deep Learning*. Cuya esencia se resume como sigue:

1. Se ha desarrollado un modelo de arquitectura, basado en una capa de presentación y otra de negocio, para poder llevar a cabo los procesos de entrenamiento y clasificación.
2. Se ha seleccionado un *dataset* apropiado con la finalidad de detectar personas bajo una perspectiva de dron.
3. Se ha configurado la tecnología *YOLO* en dos de sus variantes, así como los parámetros necesarios.
4. Se han validado los desarrollos, utilizando las métricas pertinentes.
5. Se han integrado todos los módulos que conforman la aplicación, verificando su correcto funcionamiento.

5.2. Conclusions

A year ago, when the idea of this project was proposed, there was a high level of enthusiasm among the members of the group. The three members had a great curiosity towards Artificial Intelligence, thanks to the completion of some subjects during the degree, many of them focused on machine learning. However, most of the techniques studied worked on tabular data. This is why carrying out such a large-scale project - a dissertation on the detection of people in images - was a great opportunity to learn about a variant of machine learning that, until then, had been unknown.

After completing the dissertation, the three authors agree that it has been a great experience and a very complete learning process. Carrying out this work has allowed the authors to broaden their theoretical knowledge of the underlying models and techniques. Concepts that were previously unknown, such as convolutional neural networks or clustering operations, are no longer strange and, from now on, are included in the knowledge acquired after carrying out this project. On the other hand, carrying out this work has allowed us to approach a new technology that has never been studied before, such as Matlab and its application design tool, Matlab App Designer. After its use, it is considered that this option is as valid to carry out these tasks as other, initially more appealing ones, such as Python and its associated libraries. The development team has felt comfortable with the environment from the very first moment and, thanks to a clear and abundant documentation, no inconveniences of any kind have been encountered.

It is worth mentioning that, during this period, not only theoretical foundations have been acquired. A Bachelor's Thesis, due to its complexity, requires a high level of cohesion, structure and planning on the part of the members. Everyone agrees that good organisation is fundamental for the correct execution of any activity of this kind, and that, in some cases, it is as important as the theoretical concepts to be explored. This is why, during these months, they have insisted on following guidelines to avoid last-minute surprises or lack of control.

In addition, this work has been an excellent opportunity for the team members to face in a real way the challenges and dynamics of an applied research project. From setting goals to evaluating results, they have had to deal with typical problems in developing computer vision systems, such as image processing, managing large datasets and evaluating the performance of trained models. Rather than being an insurmountable obstacle, these difficulties have been a constant source of learning, strengthening skills such as resilience, critical thinking and analytical skills.

In addition, a self-taught approach has been promoted to deepen technical aspects not directly covered in the degree, such as understanding advanced neural network architectures (such as YOLO and Faster R-CNN) or techniques to improve performance through regularisation, data augmentation or hyperparameter tuning. All this has sparked a greater interest in artificial intelligence research, opening the door to possible master's studies or even future job opportunities in this field.

Finally, it is crucial to highlight the personal impact that this Bachelor's Thesis has

had on the members of the group. Beyond the technical knowledge acquired, the project has served as a validation of their own abilities and has fostered a sense of achievement that goes beyond the academic.

For all of the above, the team is very satisfied with the work carried out and grateful for the opportunity given to them by their tutor to carry it out.

In conclusion, it is worth highlighting that all the objectives proposed at the beginning of the project have been successfully achieved. These are reflected in the developed application—an object detection system using Deep Learning techniques—summarized as follows:

1. An application architecture has been developed, based on a presentation layer and a business logic layer, to carry out training and classification processes.
2. An appropriate dataset has been selected with the aim of detecting people from a drone's perspective.
3. YOLO technology has been configured in two of its variants, along with the necessary parameters.
4. The developments have been validated using the relevant metrics.
5. All modules that make up the application have been integrated and their correct functionality verified.

5.3. Trabajo Futuro

A continuación se indican algunas consideraciones relativas a posibles mejoras de futuro, identificándose por la naturaleza de las mismas.

5.3.1. Color Quantization using K-Means

Uno de los inconvenientes que el equipo ha encontrado empleando Lacmus Drone Dataset (LADD) es que las imágenes tienen resoluciones elevadas, lo que implica que el dataset sea complicado de procesar computacionalmente hablando. Los modelos de reconocimiento de objetos requieren recorrer las imágenes en numerosas ocasiones, es por eso que el equipo de desarrollo considera que pueden emplearse técnicas de preprocesado a las imágenes con tal de mejorar el rendimiento de los entrenamientos.

La primera idea que el equipo propone es la de emplear técnicas de reducción de color sobre las imágenes. Las imágenes poseen una amplia gama de colores, derivados de la combinación de los tres canales (Rojo, verde y Azul) donde cada canal admite un rango de valores de hasta 256 niveles con una representación de 8 bits. Es por eso que se considera

que, empleando este preprocesado se puede lograr reducir el número de colores de las mismas. Al disminuir la cantidad de colores, se simplifica la representación visual de la imagen. Esto puede ayudar al modelo a enfocarse en las características estructurales y contornos en lugar de distraerse con variaciones sutiles de color que podrían considerarse ruido en determinados contextos.

El algoritmo empleado para llevar a cabo esta tarea es *K-Means*. Uno de los algoritmos de *Clustering* más eficientes y ampliamente utilizados en el mundo del *Machine Learning*. Los algoritmos de *Clustering* consisten en agrupar una serie de datos en lo que se denominan *Clusters*, agrupaciones de datos que se llevan a cabo en función de la distancia de los mismos. A continuación, se muestra un ejemplo de un espacio de dos dimensiones en el que los datos se representan mediante dos valores, X e Y, en el que se ha aplicado un algoritmo de *clustering* especificando que se dividan los datos en dos y tres *Clusters*.

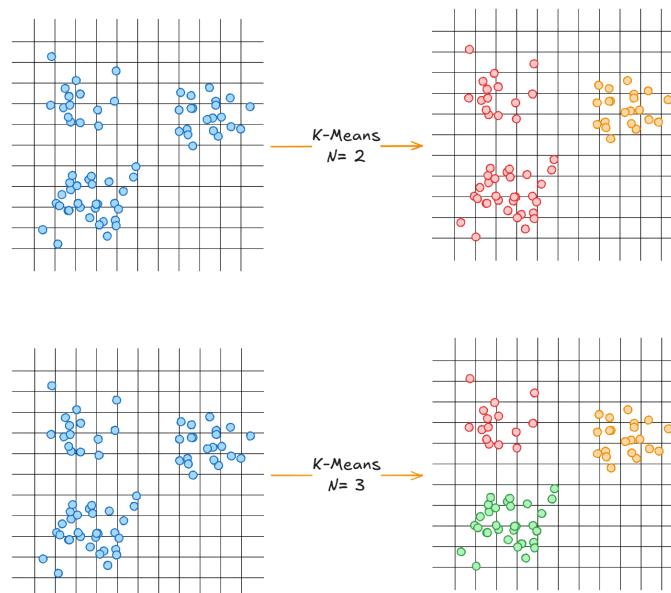


Figura 5.1: Ejemplo de aplicación de algoritmo de clustering, Fuente: Propia

Definición Simple del Algoritmo K-means

El algoritmo K-means sigue los siguientes pasos:

1. **Inicialización:** Se seleccionan K ejemplares de forma aleatoria, que denominamos centriodes.
2. **Asignación:** Cada ejemplar se asigna al centroide más cercano.
3. **Actualización:** Se recalculan los centros como la media de los puntos asignados.
4. **Repetición:** Se repiten los pasos de asignación y actualización hasta que los centros ya no cambien significativamente.

Ahora este algoritmo puede aplicarse a los colores de las imágenes. Una imagen contiene píxeles, y un píxel es una tripleta de canales de color: uno para rojo, otro para verde y otro para azul. Cada uno de los valores tiene una profundidad de 8 *bits*, lo que significa que el valor de cada color es un número comprendido entre 0 y 2^8 (256). Estos valores pueden ser representados en un espacio vectorial tridimensional, de forma que los datos a agrupar son dichas tripletas. A continuación, se muestra un ejemplo de reducción de color (Figura 5.3).

En la imagen original encontramos que la cantidad de colores únicos es de 96.615. Esto significa que hay exactamente ese número de tripletas / píxeles diferentes en la imagen. En la Figura 5.3 se observa el resultado de aplicar una reducción de color a 64 colores, esto significa que cada color de la primera imagen ha sido convertida a uno de los 64 clusters actuales, concretamente al más cercano dentro del espacio vectorial.



Figura 5.2: Imagen original

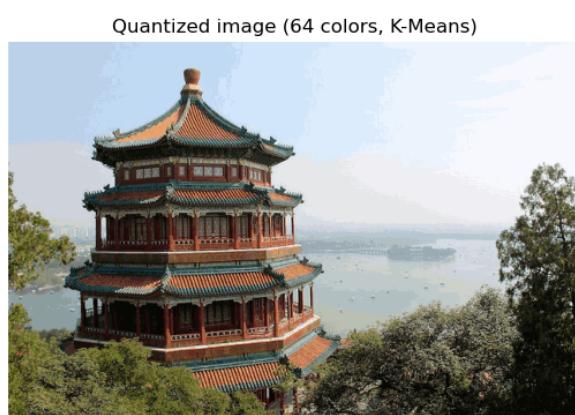


Figura 5.3: Imagen tras aplicarle K-Means

5.3.2. Detección en vídeo

Una posible ampliación de este proyecto sería aplicar la detección de personas no solo en imágenes estáticas, sino también en vídeos. Este paso permitiría usar el sistema en situaciones más reales, como vigilancia en tiempo real, análisis de comportamiento o conteo de personas en movimiento.

Para ello, el primer cambio necesario sería adaptar el sistema para procesar vídeos, ya sean grabaciones almacenadas o transmisiones en directo desde una cámara. Este proceso implicaría extraer cada fotograma del vídeo y tratarlo como una imagen individual, aplicando sobre él el mismo modelo de detección de personas desarrollado en este trabajo. Sin embargo, a diferencia de las imágenes sueltas, en vídeo es fundamental que el sistema sea capaz de realizar este análisis de forma rápida y continua para mantener un flujo fluido sin interrupciones. Para ello se podrían emplear versiones reducidas de *YOLO* como las denominadas *nano* o *tiny*. Algunos ejemplos son: *YOLOv5n*, *YOLOv8n*, *YOLOv4-tiny*, *YOLOv7-tiny*. Por tanto, sería necesario optimizar el rendimiento del modelo, posiblemente reduciendo su complejidad o utilizando versiones más ligeras.

Además de detectar personas en cada fotograma, es esencial incorporar un sistema de seguimiento (*tracking*) que permita mantener la identidad de cada persona a lo largo del tiempo. Esto significa que el sistema debe ser capaz de identificar que una persona detectada en un fotograma es la misma que aparece en los siguientes, incluso si cambia ligeramente de posición, se superpone con otra o se oculta parcialmente.

El proceso completo seguiría los siguientes pasos:

1. **Captura del vídeo:** Se extraen los fotogramas a una frecuencia determinada (por ejemplo, 15 o 30 fps¹).
2. **Detección por fotograma:** Se aplica el modelo de detección de personas en cada imagen individual.
3. **Asociación y seguimiento:** Se comparan las personas detectadas en el nuevo fotograma con las del anterior, asignando una ID única a cada una y trazando su trayectoria.
4. **Visualización y análisis:** Se muestran los resultados en tiempo real, con cajas delimitadoras e identificadores únicos, y se podrían guardar datos como número de personas, trayectorias o tiempo de permanencia en una zona.

¹FPS: *Frames Per Second* o cuadros por segundo, una medida de cuántas imágenes procesa el sistema por segundo.

5.3.3. Mejora de la detección en condiciones adversas

Uno de los principales retos en operaciones de rescate es la detección de personas en situaciones complejas, como puede ser en zonas boscosas, con poca visibilidad o condiciones climáticas adversas (nieve, niebla, lluvia). Por ello, se plantea:

- Integrar modelos de mejora de imagen, como algoritmos de aumento de contraste, reducción de niebla o mejora en condiciones de baja iluminación.
- Explorar el uso de cámaras térmicas o infrarrojas. Estas tecnologías permiten detectar fuentes de calor, como el cuerpo humano, incluso en entornos donde la visibilidad es muy reducida. La combinación de imágenes RGB² y térmicas podría generar una solución más robusta, especialmente durante operaciones nocturnas o en zonas con mucha vegetación.

5.3.4. Inclusión de sistemas de geolocalización

Para facilitar la intervención de equipos de rescate, se propone incluir la capacidad de asociar cada detección con coordenadas GPS proporcionadas por el dron.

Esta integración permitiría visualizar cada detección sobre un mapa interactivo, lo que ayudaría al equipo de rescate a planificar rutas de acceso más efectivas. A lo largo de una operación, podría generarse un mapa de calor con las zonas donde más frecuentemente se detectan personas, lo cual serviría para identificar áreas de especial interés o evaluar patrones de movimiento.

5.3.5. Automatización y sistema de alertas

Una posible evolución del sistema es la automatización del proceso de notificación.

Se podría desarrollar un sistema de alertas inteligentes que, al detectar una persona con un alto grado de confianza, emita automáticamente una notificación al equipo de rescate o seguridad de la zona.

Estas alertas podrían enviarse a través de una aplicación móvil, una interfaz web o incluso vía mensaje automático, permitiendo una reacción rápida.

²RGB: *Red, Green, Blue* (rojo, verde, azul), modelo de color aditivo utilizado en sensores de imagen, pantallas y cámaras para representar colores mediante la combinación de estos tres canales.

5.3.6. App orientada a cuerpos de emergencia

Actualmente, la aplicación está diseñada como una herramienta de entrenamiento de modelos. Una posible evolución sería el desarrollo de una versión específica para equipos de emergencia, esta nueva herramienta estaría pensada para su uso directo en situaciones reales de búsqueda y rescate. Su objetivo principal sería proporcionar a los equipos de intervención una plataforma intuitiva, ágil y confiable que facilite la toma de decisiones sobre el terreno.

La funcionalidad central de esta aplicación sería la recepción en tiempo real de las detecciones generadas por los drones desplegados. A medida que el dron sobrevuela una zona y realiza capturas de vídeo o imágenes, el modelo de detección procesaría los datos y enviaría las detecciones relevantes a la aplicación. Estas detecciones incluirían información visual, como la imagen o fotograma en el que se ha identificado una posible persona, y datos analíticos asociados como la coordenada GPS³, la hora de la detección y el nivel de confianza estimado por el modelo.

Además, la interfaz permitiría al usuario visualizar en un mapa la posición exacta de cada detección, así como la trayectoria del dron. De este modo, los operadores podrían ver no solo los puntos en los que se han realizado identificaciones, sino también zonas que ya han sido exploradas, lo cual ayuda a optimizar la cobertura del terreno. Las detecciones podrían agruparse por zonas, destacando aquellas áreas donde se han registrado múltiples indicios de presencia humana, y generando mapas de calor automáticos en función de la densidad de hallazgos.

Otro aspecto fundamental de la aplicación sería la consulta del historial de operaciones. Cada misión podría registrarse con un identificador único, incluyendo detalles como la fecha, la zona geográfica cubierta, el número de detecciones, el tiempo de vuelo y el recorrido realizado. Esto permitiría no solo analizar el rendimiento del sistema, sino también reutilizar información en futuras misiones, por ejemplo, para priorizar áreas que históricamente han presentado mayor riesgo o donde ha habido mayor actividad humana.

La aplicación también podría integrar filtros para revisar únicamente detecciones con un nivel alto de confianza, marcar detecciones revisadas por un operador humano, o establecer niveles de criticidad dependiendo del contexto (por ejemplo, detecciones en zonas de difícil acceso o en condiciones climáticas extremas).

³GPS: *Global Positioning System* (Sistema de Posicionamiento Global), tecnología de navegación satelital que permite determinar la ubicación geográfica de un receptor en la Tierra con alta precisión.

Contribuciones Personales

Ignacio García Fernández

Labores de investigación:

- Investigación exhaustiva de los requisitos necesarios para encontrar un conjunto de datos (*DataSet*) adecuado, que cumpla con las especificaciones técnicas y licencias necesarias para el proyecto.
- Estudio y familiarización con el entorno de programación Matlab y su lenguaje propio, comenzando desde cero sin conocimientos previos, para garantizar un buen manejo durante el desarrollo.
- Profundización en el funcionamiento y entrenamiento de redes neuronales convolucionales, con especial atención a los modelos Yolov4 y YoloX, para entender sus capacidades y limitaciones.
- Análisis detallado de la herramienta Matlab App Designer, centrándose en sus capacidades para el desarrollo de interfaces gráficas orientadas a facilitar la interacción con la aplicación.
- Investigación sobre la creación y funcionalidad de elementos gráficos interactivos como botones, campos de texto e imágenes, así como la depuración y manejo de variables dentro de App Designer.
- Diseño conceptual y elaboración de esquemas para la interfaz gráfica de usuario, asegurando una estructura clara y amigable que facilite la usabilidad.
- Estudio comparativo y selección de métodos de optimización (optimizers o solvers), evaluando especialmente Adam y sgdm, para optimizar el proceso de entrenamiento de las redes neuronales.

Labores de gestión:

- Creación y mantenimiento de un repositorio en GitHub para almacenar de manera organizada y segura el código fuente de la aplicación.
- Participación en la elección y definición de la metodología de trabajo a emplear durante el desarrollo del proyecto, buscando maximizar la eficiencia y coordinación del equipo.
- Planificación y reparto equilibrado de tareas entre los miembros del equipo, estableciendo fechas límite realistas para la entrega de cada fase.
- Revisión crítica y corrección de errores en las diferentes versiones de la memoria del proyecto para mejorar la calidad del documento final.

Labores de documentación:

- Realización de los apartados 1.1, 1.2, 1.3, 1.4 de la introducción en los que se detallan los antecedentes, la motivación del proyecto, los objetivos y el plan de trabajo
- Realización de los puntos 2.1, 2.2 y 2.3 en los que se da una introducción conceptual a las operaciones subyacentes a los modelos, a los tensores y a los métodos de optimización.
- Realización del punto 3.1 en el que se detalla la arquitectura planteada para la aplicación, presentando sus capas con detalle.
- Realización del punto 3.2 donde se explican los casos de usos realizados durante el proyecto desde un punto de vista de la ingeniería del *software*.
- Realización del apartado 3.3 donde se da una primera presentación a la aplicación desarrollada. Detallando los primeros componentes con los que el usuario final debe interactuar
- Realización del apartado 4.1.1 en el que se da una primera definición el *dataset* empleado, qué tipo de imágenes contiene, de qué tipo y la licencia asociada al mismo.
- Redacción del punto 4.2.3 en el que se presentan los resultados obtenidos por las diferentes variantes de los modelos entrenados tanto en la fase de entrenamiento como en la de test, y mencionando los modelos que mejores resultados han obtenido.
- Redacción del punto 5.3.1 de trabajo futuro en el que se menciona la posibilidad de emplear la técnica de reducción de color en las imágenes, empleando el algoritmo de *clustering K.means*. Con el fin de facilitar la generalización de los modelos.

Jorge Serrano Velázquez

Labores de investigación:

- Investigación exhaustiva de los requisitos necesarios para encontrar un conjunto de datos (*DataSet*) adecuado, que cumpla con las especificaciones técnicas y licencias necesarias para el proyecto.
- Revisión de trabajos académicos y proyectos existentes relacionados con el reconocimiento de personas mediante imágenes captadas por drones, con el objetivo de identificar enfoques, metodologías y posibles mejoras aplicables al presente proyecto.
- Estudio y familiarización con el entorno de programación Matlab y su lenguaje propio, comenzando desde cero sin conocimientos previos, para garantizar un buen manejo durante el desarrollo.
- Diseño conceptual y elaboración de esquemas para la interfaz gráfica de usuario, asegurando una estructura clara y amigable que facilite la usabilidad.
- Estudio de técnicas de preprocesamiento de imágenes para mejorar la calidad de los datos utilizados.

Labores de desarrollo:

- Búsqueda y selección de *datasets* específicos que se ajustaran a las necesidades del proyecto, priorizando calidad, variedad y licencia adecuada para su uso.
- Colaboración activa en el diseño de la interfaz de la aplicación, aportando ideas y mejoras junto al equipo para mejorar la experiencia del usuario.
- Ejecución de múltiples entrenamientos con los modelos *YOLOv4* y *YOLOX*, explorando distintas configuraciones de parámetros y optimizadores para mejorar el rendimiento.
- Clasificación de imágenes utilizando *YOLOv4* y realización de comparativas detalladas con los resultados obtenidos con *YOLOX* para seleccionar el mejor modelo.
- Desarrollo integral del código fuente y funcionalidades de la aplicación, incluyendo módulos para la clasificación, entrenamiento y procesamiento de imágenes.
- Integración del código desarrollado con las funciones específicas necesarias para cada tarea, asegurando un sistema cohesivo y eficiente.
- Creación de un *script* en Bash para programar los entrenamientos de los modelos y evitar lanzar dos entrenamientos simultáneamente y permitiendo ver su progreso y resultado desde un *bot* de *Telegram*.

Labores de gestión:

- Organización y administración del espacio de trabajo en *Google Drive* para el equipo, facilitando la colaboración y el acceso a los documentos y recursos del proyecto.
- Detección de problemas técnicos y dudas surgidas durante el desarrollo, preparándolas para su discusión y resolución en reuniones con el equipo docente.
- Planificación y moderación de las reuniones semanales de seguimiento, siguiendo los principios de la ingeniería del *software* para garantizar una comunicación efectiva y el avance continuo del proyecto.
- Detección de problemas técnicos y dudas surgidas durante el desarrollo, preparándolas para su discusión y resolución en reuniones con el equipo docente.

Labores de documentación:

- Redacción de los apartados 1.5 y 1.6 en los que se expone el alcance y limitaciones del proyecto, así como la organización general de la memoria, estructurando el contenido de forma coherente y accesible.
- Redacción de los puntos 1.11, 1.12 y 1.13 en inglés, correspondientes al plan de trabajo, alcance y limitaciones, y la organización de la tesis respectivamente, adaptando el contenido al idioma y estilo técnico requerido.
- Redacción del apartado 3.6.1 sobre la metodología utilizada para la gestión del proyecto.
- Desarrollo del punto 3.6.2, donde se explica la organización del trabajo mediante *User Story Map* e historias de usuario.
- Documentación del apartado 3.6.3, centrado en la gestión del trabajo en curso (*WIP*).
- Redacción del punto 3.6.4, en el que se describen los distintos tipos de reuniones llevadas a cabo.
- Elaboración del apartado 3.6.5, donde se identifican y explican los hitos más relevantes del proyecto.
- Desarrollo del punto 3.6.6, centrado en la gestión de configuración del *software* y las herramientas empleadas.
- Redacción del punto 4.1, la descripción de los recursos disponibles, 4.1.2 el análisis detallado del *dataset* utilizado, 4.1.3 todo el *software* usado en el proyecto y 4.1.4 las características del *hardware* empleadas para el correcto desarrollo del proyecto.
- Elaboración del apartado 4.2.2 en el que se definen las métricas utilizadas para la evaluación del desempeño del sistema desarrollado.
- Redacción de los puntos 5.3.3, 5.3.4 y 5.3.5 correspondientes al trabajo futuro, en los que se plantean posibles mejoras en la detección en condiciones adversas, la integración de sistemas de geolocalización y el desarrollo de sistemas de alertas automáticas.

Jorge Torres Ruiz

Labores de investigación:

- Estudio y familiarización con el entorno de programación Matlab y su lenguaje propio, comenzando desde cero sin conocimientos previos, para garantizar un buen manejo durante el desarrollo.
- Estudio detallado y comprensión profunda del código guía proporcionado para entender y replicar las funcionalidades implementadas en el proyecto.
- Análisis de las diferentes opciones de entrenamiento y métodos de optimización utilizados en el proyecto para seleccionar los más adecuados.
- Evaluación de Python como posible lenguaje alternativo para el desarrollo de la aplicación, considerando ventajas y desventajas frente a Matlab.
- Profundización en el funcionamiento y entrenamiento de redes neuronales convolucionales, con especial atención al modelo Yolov4, para entender sus capacidades y limitaciones.

Labores de desarrollo:

- Ejecución de múltiples entrenamientos con los modelos Yolov4, explorando distintas configuraciones de parámetros y optimizadores para mejorar el rendimiento.
- Realización de pruebas exhaustivas para validar el correcto funcionamiento de todas las funcionalidades implementadas en la aplicación.
- Diseño y creación de las distintas vistas o pantallas de la aplicación, incluyendo la página principal, sección de ayuda, y ventanas específicas para entrenamiento y clasificación.
- Desarrollo integral del código fuente y funcionalidades de la aplicación, incluyendo módulos para la clasificación, entrenamiento y procesamiento de imágenes.
- Integración del código de la aplicación con cada una de las funciones desarrolladas y utilizadas para cada función específica.

Labores de documentación:

- Realización de los apartados 1.7, 1.8, 1.9 y 1.10 de la introducción, donde se detallan en inglés los antecedentes, motivación y objetivos.
- Realización del apartado 2.4, apartado en el que se explican algunas de las funciones de activación más importantes de las redes neuronales, como pueden ser las funciones *Sigmoide*, *ReLU*, *LeakyReLU* y *SoftMax*.

- Realización del apartado 2.5, donde se explica qué son las Redes neuronales convolucionales (CNN) y se da una visión detallada de las operaciones y técnicas presentes en sus estructuras, como pueden ser la operación de convolución, el agrupamiento (*Pooling*), el sobreajuste, la normalización, el *Mean Squared Error* o el *Cross-Entropy Loss*.
- Realización del apartado 2.6 en el que se detalla la detección de objetos en imágenes, más concretamente con *YOLO* y algunas de sus versiones como *YOLOv4* y *YOLOX*.
- Realización de los apartados 3.4 y 3.5 donde se describen las fases de entrenamiento y clasificación del modelo, permitiendo configurar hiperparámetros como el optimizador, épocas y tasa de aprendizaje, así como ajustar la detección de humanos mediante umbrales y visualizar resultados con métricas de precisión.
- Realización de los apartados 5.1 y 5.2 donde se indican las conclusiones del proyecto, tanto en español como en inglés.
- Realización del apartado 5.3.2 del Trabajo a futuro, donde se indican ampliaciones del proyecto para analizar vídeos en tiempo real.
- Realización del apartado 5.3.6 del Trabajo a futuro, donde se indican ampliaciones del proyecto para realizar una versión especializada para equipos de rescate, integrando detecciones en tiempo real desde drones con datos GPS y optimizando búsquedas con mapas de calor entre otras cosas.

Control de calidad:

- Realización de entrenamientos con los modelos Yolov4 y YoloX, buscando ajustar parámetros para conseguir resultados óptimos y robustos.
- Ejecución de procesos de clasificación de imágenes y verificación del correcto funcionamiento y precisión en las predicciones de los modelos entrenados.
- Pruebas específicas para la identificación de objetos usando Yolov4 y YoloX, garantizando la fiabilidad de la aplicación.
- Análisis comparativo de los resultados obtenidos con los diferentes modelos, incluyendo puesta en común de hallazgos y discusión de mejoras.
- Testeo final y validación de todas las funcionalidades desarrolladas para asegurar su correcto desempeño.
- Seguimiento constante y documentación de los problemas detectados durante el desarrollo y las soluciones aplicadas para mejorar la calidad del proyecto.

Bibliografía

- [1] MathWorks, “Getting started with yolox object detection,” <https://es.mathworks.com/help/vision/ug/getting-started-with-yolox-object-detection.html>.
- [2] GREIM, “Greim - grupos de rescate especial de intervención en montaña,” Disponible en <https://www.guardiacivil.es/es/institucional/Conocenos/especialidades/Greim/index.html> [Acceso Feb. 2025].
- [3] M. F. Ahmed, “The performance assessment of reverse osmosis stations at al-mahalabea area,” *Scientific Review Engineering and Environmental Sciences*, vol. 29, no. 3, pp. 332–342, 2020. [Online]. Available: https://www.researchgate.net/publication/344434297_The_performance_assessment_of_reverse_osmosis_stations_at_Al-Mahalabea_area
- [4] N. Shen, L. Chen, J. Liu, L. Wang, T. Tao, D. Wu, and R. Chen, “A review of global navigation satellite system (gnss)-based dynamic monitoring technologies for structural health monitoring,” *Remote Sensing*, vol. 11, no. 9, p. 1001, 2019. [Online]. Available: <https://www.mdpi.com/2072-4292/11/9/1001>
- [5] B. Arbanas, F. Petric, A. Batinović, M. Polić, I. Vatavuk, L. Marković, M. Car, I. Hrabar, A. Ivanović, and S. Bogdan, *From ERL to MBZIRC: Development of An Aerial-Ground Robotic Team for Search and Rescue*. IntechOpen, 2021. [Online]. Available: <https://www.intechopen.com/chapters/77963>
- [6] P. N. de Ordesa y Monte Perdido, “Parque nacional de ordesa y monte perdido - información oficial,” Disponible en <https://www.miteco.gob.es/es/red-parques-nacionales/nuestros-parques/ordesa/> [Acceso Mayo 2025].
- [7] European Union Aviation Safety Agency, “European union aviation safety agency (easa),” <https://www.easa.europa.eu/en>, 2025.
- [8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, disponible en: <https://www.deeplearningbook.org/> [Acceso Mayo 2025]. [Online]. Available: <https://www.deeplearningbook.org/>
- [9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986, disponible en: <https://www.nature.com/articles/323533a0> [Acceso Mayo 2025].

- [10] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: MIT Press, 2012, disponible en: <https://mitpress.mit.edu/9780262018029> [Acceso Mayo 2025].
- [11] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, vol. 28, 2013, pp. 1139–1147, disponible en: <https://proceedings.mlr.press/v28/sutskever13.html> [Acceso Mayo 2025].
- [12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS’12)*, vol. 1. USA: Curran Associates Inc., 2012, pp. 1097–1105, disponible en: https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html [Acceso: Mayo 2025].
- [13] DataCamp, “Yolo object detection explained,” Disponible en <https://www.datacamp.com/es/blog/yolo-object-detection-explained> [Acceso Mayo 2025].
- [14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 779–788, disponible en: <https://arxiv.org/abs/1506.02640>.
- [15] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” 2020, disponible en: <https://arxiv.org/abs/2004.10934>.
- [16] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, “YOLOX: Exceeding YOLO Series in 2021,” *arXiv preprint arXiv:2107.08430*, 2021, disponible en: <https://arxiv.org/abs/2107.08430> [Acceso Enero 2025].
- [17] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755, disponible en: <https://cocodataset.org>.
- [18] M. Shuranov, D. Shurenkov, D. Ruzhitsky, V. Martynova, E. Bykova, and G. Pervezchikov, “Ladd: Lacmus drone dataset,” Disponible en <https://datasetninja.com/lacmus-drone-dataset> [Acceso Feb. 2025], 2023.
- [19] Kaggle, “Kaggle - plataforma de datasets,” Disponible en <https://www.kaggle.com> [Acceso Feb. 2025].
- [20] D. Ninja, “Dataset ninja - plataforma de datasets,” Disponible en <https://datasetninja.com/> [Acceso Feb. 2025].

Apéndice: Manual de Usuario

A continuación se explican los pasos detallados que deben realizarse para ejecutar correctamente la aplicación:

1. Descargar el código de la aplicación ubicado en el siguiente repositorio alojado en GitHub: <https://github.com/G2simp/TFG>
2. Es necesario disponer del dataset utilizado para el entrenamiento de las redes neuronales. Lacmus Drone Dataset (LADD) [18].
3. Descargar Matlab R2024b necesario para ejecutar la aplicación: https://es.mathworks.com/products/new_products/release2024b.html
4. Instalar Matlab junto a los siguientes add-ons, que configuran los toolboxes necesarios para el desarrollo y ejecución de la aplicación:
 - i. Computer Vision Toolbox
 - ii. Automated Visual Inspection Library for Computer Vision Toolbox
 - iii. Computer Vision Toolbox Model for YOLO v4 Object Detection
 - iv. Curve Fitting Toolbox
 - v. Deep Learning Toolbox
 - vi. Deep Learning HDL Toolbox
 - vii. Image Processing Toolbox
 - viii. Parallel Computing Toolbox
 - ix. Statistics and Machine Learning Toolbox

5. Una vez instalados todos los recursos señalados, abrir la aplicación y situarse en la carpeta del código descargado en el primer paso. Hacer doble clic izquierdo o clic derecho → Open sobre el archivo `app2.mlapp` que aparece en el panel izquierdo.

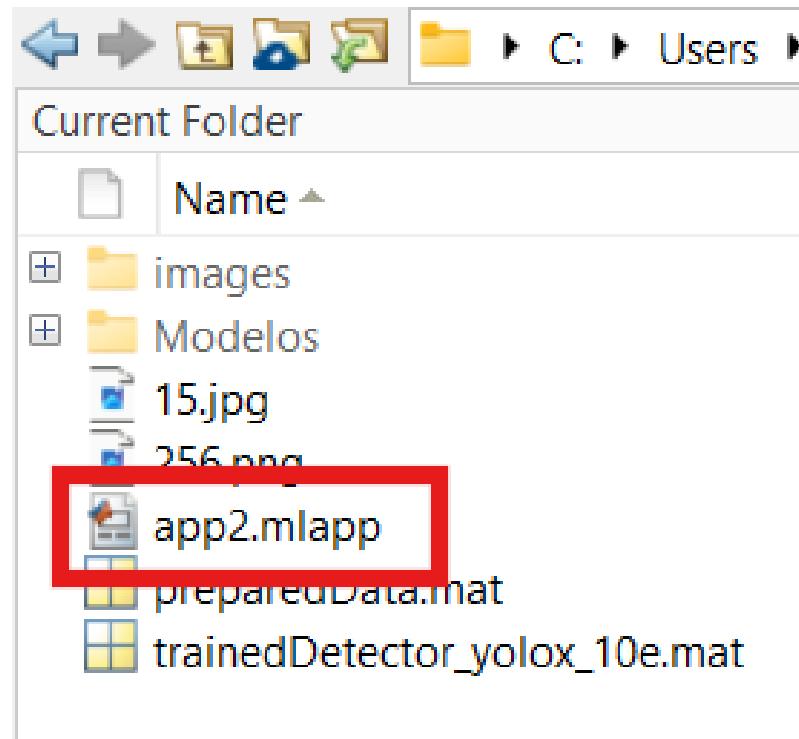


Figura 4: Panel Matlab

6. Se abrirá la vista de Matlab App Designer, con una vista previa estática de la aplicación. Pulsar sobre el botón verde *Run* del panel superior.

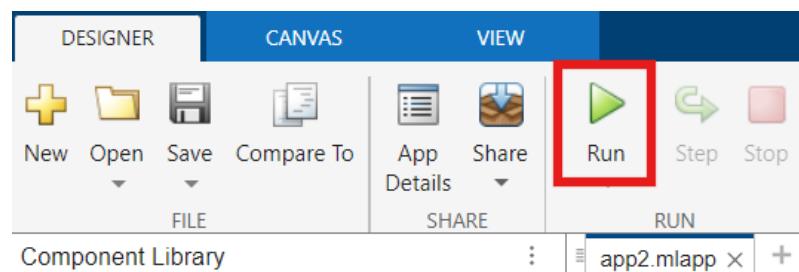


Figura 5: Botón Run

7. La ventana con la aplicación se abrirá, mostrándose la vista principal, lista para usarse.