

Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Visión artificial en la identificación de drones

Grado en Ingeniería Mecánica

ALUMNO: Álvaro Tortosa Montalvo

DIRECTOR: José Ignacio Valles Cancela

CURSO ACADÉMICO: 2025-2026

Universidade Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Visión artificial en la identificación de drones

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval

Cuerpo General

Universidade Vigo

RESUMEN

El uso masivo de sistemas aéreos no tripulados y su papel decisivo en conflictos contemporáneos ha evidenciado la necesidad de desarrollar sistemas de vigilancia más robustos. En este contexto, el presente Trabajo Fin de Grado propone una solución de visión artificial orientada a la clasificación entre drones y aves.

Para ello, se ha diseñado una red neuronal convolucional desde cero, que se ha mejorado progresivamente mediante un proceso de prueba y error hasta obtener una arquitectura final optimizada. Se ha llevado a cabo un estudio comparativo de tres modelos distintos: un modelo inicial, un segundo al que se le ha incorporado una capa de extracción de bordes con el objetivo de mejorar el aprendizaje y, por último, un modelo basado en *transfer-learning* con *fine-tuning*.

Para la evaluación, se han utilizado tanto métricas de rendimiento como herramientas de interpretación, como los mapas de calor Grad-CAM y las gráficas de entrenamiento, lo que ha permitido analizar el comportamiento de los modelos y su capacidad de generalización.

Los resultados obtenidos muestran que los modelos desarrollados alcanzan un buen rendimiento en la tarea de clasificación, reforzando la viabilidad del uso de técnicas de *deep learning* para la clasificación de drones, y sirviendo como base para futuras mejoras y ampliaciones del sistema.

PALABRAS CLAVE

Dron, Pájaro, Clasificación, Visión Artificial, Red Neuronal Convolucional (CNN)

AGRADECIMIENTOS

Me gustaría expresar mi más sincero agradecimiento a mi tutor, el Capitán de Corbeta José Ignacio Valles Cancela, por haberme enseñado desde cero y con paciencia todo lo necesario para el desarrollo de este trabajo. Además, me gustaría dejar constancia del esfuerzo y el tiempo que ha dedicado al mismo, siendo fundamental para alcanzar el objetivo final y hacer de este trabajo una realidad.

Quiero agradecer también a mi familia, en especial a mis padres, por su apoyo constante a lo largo de estos años.

A mis amigos de la Escuela Naval Militar, con quienes he compartido estos años de formación y muchos momentos de diversión.

Por último, a todos aquellos que, en mayor o menor medida y de una forma u otra me ayudaron en diferentes fases de este trabajo, mostrando su interés y disponibilidad.

CONTENIDO

Contenido 1
Índice de Figuras 5
Índice de Tablas 7
1 Introducción y objetivos 9
1.1 Introducción y Motivación 9
1.2 Objetivos 10
1.3 Estructura 11
2 Estado del arte 13
2.1 Contextualización 13
2.2 Problemática de la identificación de drones 13
2.2.1 Aparición masiva de drones comerciales 13
2.2.2 Riesgos en entornos civiles y militares 14
2.2.3 Dificultad de distinguir drones de aves con sensores tradicionales 15
2.3 Otros medios de detección y clasificación 16
2.4 Visión Artificial aplicada a la detección y clasificación de objetos 18
2.4.1 Definición de Visión Artificial 18
2.4.2 Detección y clasificación de objetos 19
2.4.3 Aplicaciones en vigilancia y seguridad 19
2.4.4 Evolución desde métodos clásicos a *Deep Learning* 20
2.5 Redes Neuronales Convolucionales (CNN) 21
2.5.1 Concepto de Redes Neuronales Convolucionales (CNN) 21
2.5.2 Redes Neuronales Convolucionales en el procesamiento de imágenes 22
2.5.3 Ventajas frente a métodos tradicionales 22
2.6 Modelos actuales de detección en tiempo real 23
2.6.1 La familia R-CNN: detección en dos etapas 23
2.6.2 SSD (*Single Shot MultiBox Detector*) 23
2.6.3 YOLO (*You Only Look Once*) 24
2.6.4 *Vision Transformers* 24
2.7 Uso de *datasets* abiertos y entrenamiento supervisado 25
2.7.1 Importancia de los *datasets* etiquetados 25
2.7.2 Limitaciones de los *datasets* genéricos 26
2.7.3 Necesidad de datos reales del entorno 26
3 Desarrollo del TFG 29
3.1 Preparación del *dataset* 29

3.1.1 Origen del *dataset* 29
3.1.2 Filtrado 29
3.1.3 Particionado del *dataset* 30
3.2 Metodología 31
3.2.1 Papel del *dataset* en el proceso de trabajo 31
3.2.2 Optimización del modelo y transformaciones al *dataset* 31
3.2.3 Optimización del bucle de entrenamiento 34
3.3 Arquitectura de la red 34
3.3.1 Introducción 34
3.3.2 Inicialización y entorno de ejecución 35
3.3.3 Capa de extracción de bordes 36
3.3.4 Arquitectura principal *DeepCNN* 37
3.3.5 Resumen técnico de parámetros 43
3.3.6 Proceso de carga y preparación de datos de entrada 44
3.4 Transformaciones aplicadas a los datos 44
3.5 Configuración del entrenamiento 46
3.5.1 Optimizador del entrenamiento 46
3.5.2 Criterio de finalización del entrenamiento 47
3.5.3 Desarrollo del proceso de entrenamiento 47
3.5.4 Almacenamiento y carga del modelo entrenado 48
4 Resultados / Validación / Prueba 49
4.1 Metodología de evaluación del modelo 49
4.1.1 Evaluación del proceso de entrenamiento (*loss* y *accuracy*) 49
4.1.2 Análisis cualitativo mediante mapas de activación (Grad-CAM) 49
4.1.3 Evaluación cuantitativa mediante matriz de confusión 50
4.2 Modelos evaluados sobre el *dataset* de prueba 50
4.2.1 Modelo sin Sobel 51
4.2.2 Modelo con Sobel 54
4.2.3 Modelo *Transfer-learning* 57
4.3 Prueba con imágenes propias 59
4.3.1 Modelo sin Sobel 59
4.3.2 Modelo con Sobel 62
4.3.3 Modelo *Transfer-learning* 65
4.4 Discusión de resultados 66
5 Conclusiones y líneas futuras 69
5.1 Conclusiones 69

5.2 Líneas futuras 70
6 Bibliografía 73
Anexo I: Implicaciones Sociales, y/o Económicas, y/o Ambientales 77
Anexo II: Reflexiones Éticas y Sociales 78
Anexo III: Fuentes empleadas en la elaboración del *dataset* 79
Anexo IV: Esquema de bloques CNN 80
Anexo V: Tablas resumen técnico de parámetros de ambos modelos 82
Anexo VI: Uso de IAG 84

ÍNDICE DE FIGURAS

Figura 2-1 Diagrama de campos de aplicación de drones en el ámbito profesional [4] 14

Figura 2-2 Dron kamikaze Shahed 136 iraní [9].....15

Figura 2-3 Dron y pájaro en misma imagen (fuente: propia).....16

Figura 2-4 Sistema AARTOS de detección de drones basado en múltiples sensores [13].....17

Figura 2-5 Flujo de trabajo típico en un sistema de visión artificial, desde la adquisición de la imagen hasta la interpretación de la información [16] 18

Figura 2-6 Ejemplo de arquitectura de una red neuronal convolucional (CNN) [24].....22

Figura 2-7 Comparación entre la arquitectura de una red neuronal convolucional (CNN) y un *Vision Transformer* (ViT) [39].....25

Figura 3-1 Imagen Grad-CAM de ave (fuente: propia) 32

Figura 3-2 Imagen Grad-CAM de dron 1 (fuente: elaboración propia a partir de imagen tomada de [45]).....32

Figura 3-3 Imagen Grad-CAM de dron 2 (fuente: elaboración propia a partir de imagen tomada de [45]).....33

Figura 3-4 Errores detectados en la metodología (fuente: propia).....33

Figura 3-5 Concatenación del canal de bordes con la imagen RGB para generar una entrada de cuatro canales (fuente: propia).....37

Figura 3-6 Primer bloque convolucional de la red con capas Conv, BatchNorm, ReLU y MaxPooling (fuente: propia).....38

Figura 3-7 Primer bloque tipo *Inception* con convoluciones en paralelo y concatenación de características (fuente: propia).....39

Figura 3-8 Segundo bloque *Inception* con convoluciones en paralelo y concatenación de características (fuente: propia).....39

Figura 3-9 Convoluciones paralelas con diferentes dilataciones y concatenación de características (fuente: propia).....40

Figura 3-10 Operación de *Global Average Pooling* seguida de *flatten* (fuente: propia) 41

Figura 3-11 Bloque residual con *skip connection* y activación *ReLU* (fuente: propia) 41

Figura 3-12 Etapas finales de la red: *pooling*, convolución, *dropout* espacial y *Global Average Pooling* (fuente: propia).....42

Figura 3-13 Concatenación de los vectores de características antes de la capa clasificadora (fuente: propia).....43

Figura 3-14 Capas finales de clasificación con capas lineales y *dropout* (fuente: propia) 43

Figura 3-15 Ejemplo de lote de entrenamiento tras la aplicación de transformaciones y desnormalización para su visualización (fuente: propia).....46

Figura 4-1 Evolución de la pérdida (izquierda) y precisión (derecha) del modelo sin Sobel durante el entrenamiento (fuente: propia).....51

Figura 4-2 Ejemplos de mapas Grad-CAM generados por el modelo sin Sobel. De izquierda a derecha: imagen original, mapa de activación para la clase ave y mapa de activación para la clase dron (fuente: propia).....52

Figura 4-3 Matriz de confusión del modelo sin filtro Sobel (fuente: propia) 53

Figura 4-4 Evolución de la pérdida (izquierda) y precisión (derecha) del modelo con Sobel durante el entrenamiento (fuente: propia).....54

Figura 4-5 Ejemplos de mapas Grad-CAM generados por el modelo con Sobel. De izquierda a derecha: imagen original, mapa de activación para la clase ave y mapa de activación para la clase dron (fuente: propia).....55

Figura 4-6 Matriz de confusión del modelo con filtro Sobel (fuente: propia) 56

Figura 4-7 Evolución de la pérdida (izquierda) y precisión (derecha) del modelo *transfer-learning* durante el entrenamiento (fuente: propia).....57

Figura 4-8 Ejemplos de mapas Grad-CAM generados por el modelo transfer-learning. De izquierda a derecha: imagen original, mapa de activación para la clase ave y mapa de activación para la clase dron (fuente: propia).....58

Figura 4-9 Matriz de confusión del modelo *transfer-learning* (fuente: propia).....58

Figura 4-10 Ejemplos de mapas Grad-CAM generados por el modelo sin Sobel a imágenes propias. De izquierda a derecha: imagen original, mapa de activación para la clase ave y mapa de activación para la clase dron (fuente: propia) 60

Figura 4-11 Matriz de confusión del modelo sin filtro Sobel para prueba con imágenes propias (fuente: propia).....61

Figura 4-12 Ejemplos de mapas Grad-CAM generados por el modelo con Sobel a imágenes propias. De izquierda a derecha: imagen original, mapa de activación para la clase ave y mapa de activación para la clase dron (fuente: propia) 63

Figura 4-13 Matriz de confusión del modelo con filtro Sobel para prueba con imágenes propias (fuente: propia).....64

Figura 4-14 Ejemplos de mapas Grad-CAM generados por el modelo *transfer-learning* a imágenes propias. De izquierda a derecha: imagen original, mapa de activación para la clase ave y mapa de activación para la clase dron (fuente: propia) 65

Figura 4-15 Matriz de confusión del modelo *transfer-learning* para prueba con imágenes propias (fuente: propia).....66

Figura 4-16 Comparación de mapas Grad-CAM entre el modelo con Sobel (izquierda) y el modelo con *transfer-learning* (derecha) (fuente: propia) 68

ÍNDICE DE TABLAS

Tabla 4-1 Métricas de clasificación del modelo sin Sobel 54
Tabla 4-2 Métricas de clasificación del modelo sin Sobel 57
Tabla 4-3 Métricas de clasificación del modelo *transfer-learning* 59
Tabla 4-4 Métricas de clasificación del modelo sin Sobel para prueba con imágenes propias 62
Tabla 4-5 Métricas de clasificación del modelo con Sobel para prueba con imágenes propias 65
Tabla 4-6 Métricas de clasificación del modelo *transfer-learning* para prueba con imágenes propias 66
Tabla 4-7 Comparación de la precisión de los diferentes modelos en el *dataset* de prueba y en pruebas reales 67

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Introducción y Motivación

La creciente presencia de vehículos aéreos no tripulados, comúnmente conocidos como drones, ha introducido nuevos desafíos en los sistemas modernos de vigilancia y control del espacio aéreo. Estos dispositivos, cada vez más accesibles y fáciles de operar, pueden aparecer en entornos sensibles como aeropuertos, infraestructuras críticas o eventos multitudinarios, donde resulta fundamental identificar rápidamente su presencia y determinar su naturaleza.

En el ámbito de la defensa, diversos organismos y responsables militares han señalado la creciente relevancia de los sistemas no tripulados en los escenarios operativos futuros. En el caso de España, el Almirante Jefe del Estado Mayor de la Armada (AJEMA) ha señalado la importancia de continuar incorporando tecnologías emergentes, entre ellas los sistemas no tripulados, como parte fundamental de la evolución de las capacidades militares en las próximas décadas [1].

Los conflictos recientes han puesto de manifiesto de forma clara la creciente relevancia de los drones en los escenarios operativos actuales. En particular, la guerra entre Rusia y Ucrania ha evidenciado el papel central que estos sistemas desempeñan tanto en tareas de reconocimiento como en operaciones ofensivas. Drones comerciales modificados, plataformas de vigilancia y sistemas de ataque de bajo coste han sido utilizados de forma masiva para obtener información en tiempo real del campo de batalla, corregir fuego de artillería o atacar objetivos puntuales.

A esta tendencia se suma el desarrollo y uso creciente de municiones merodeadoras, sistemas no tripulados capaces de permanecer en vuelo hasta localizar un objetivo y atacar directamente contra él. Este tipo de plataformas, conocidas también como “drones kamikaze”, han comenzado a ser empleadas por distintos países en conflictos recientes, incluyendo desarrollos inspirados en plataformas utilizadas previamente en otros conflictos [2]. La proliferación de este tipo de sistemas refuerza la necesidad de disponer de tecnologías capaces de detectar e identificar de forma temprana objetos aéreos de pequeño tamaño, cuya presencia puede representar una amenaza significativa incluso cuando operan de forma individual o en grandes grupos.

Uno de los principales retos asociados a esta problemática es la dificultad de diferenciar de forma fiable un dron de otros elementos presentes en el entorno aéreo, especialmente de las aves. En determinadas condiciones de observación, ambos pueden presentar tamaños aparentes, formas o patrones de vuelo similares, lo que dificulta su identificación tanto para observadores humanos como para sistemas automáticos de detección.

En este contexto, la capacidad de detectar e identificar drones de manera fiable se ha convertido en un problema tecnológico relevante. Aunque existen distintos sensores y sistemas diseñados para detectar

objetos en el espacio aéreo, muchos de ellos presentan dificultades cuando se trata de distinguir entre drones y otros elementos de tamaño y comportamiento similar.

En los últimos años, el desarrollo de técnicas de visión artificial y aprendizaje profundo (*Deep Learning*) ha abierto nuevas posibilidades para abordar este tipo de problemas. Los modelos basados en redes neuronales convolucionales han demostrado una gran capacidad para analizar imágenes y extraer patrones visuales complejos, lo que los convierte en herramientas especialmente adecuadas para tareas de clasificación de objetos.

A pesar de los avances recientes en visión artificial, la diferenciación entre drones y aves continúa representando un desafío significativo. La gran variabilidad en la apariencia de ambos objetos, junto con factores como la distancia al objetivo, las condiciones de iluminación o la presencia de fondos complejos, puede dificultar la identificación correcta del objeto presente en la imagen. Estas circunstancias obligan a desarrollar modelos capaces de aprender características visuales robustas que permitan distinguir entre ambas categorías en escenarios diversos.

En este contexto, el presente Trabajo Fin de Grado se centra en el desarrollo y evaluación de un modelo de clasificación basado en redes neuronales convolucionales capaz de distinguir entre drones y aves a partir de imágenes. Para ello, se emplea un *dataset* compuesto por imágenes procedentes de diferentes fuentes públicas, complementadas con fotografías obtenidas en condiciones reales, con el objetivo de analizar la capacidad del modelo para aprender patrones visuales discriminativos y evaluar su rendimiento en la tarea de clasificación planteada.

1.2 Objetivos

El objetivo principal de este Trabajo Fin de Grado es desarrollar y evaluar un modelo de clasificación basado en redes neuronales convolucionales capaz de diferenciar entre drones y aves a partir de imágenes. Este problema resulta especialmente relevante en sistemas de vigilancia y control del espacio aéreo, donde la identificación fiable de objetos aéreos de pequeño tamaño constituye un desafío técnico importante.

Para alcanzar este objetivo general, se plantean los siguientes objetivos específicos:

• Analizar el problema de la detección e identificación de drones, revisando las tecnologías existentes y el papel de la visión artificial en este ámbito.

• Construir un *dataset* de entrenamiento adecuado para la tarea de clasificación, combinando imágenes procedentes de diferentes fuentes públicas con el fin de obtener un conjunto de datos representativo.

• Diseñar e implementar una arquitectura de red neuronal convolucional desde cero, adaptada al problema de clasificación entre drones y aves.

• Configurar el proceso de entrenamiento del modelo, aplicando técnicas de aumento de datos, optimización y regularización para mejorar la capacidad de generalización.

• Evaluar el rendimiento del modelo entrenado, utilizando métricas de clasificación y diferentes métodos de análisis para estudiar su comportamiento.

Este enfoque permite estudiar la capacidad de los modelos basados en aprendizaje profundo (*Deep Learning*) para aprender características visuales discriminativas entre ambas categorías y valorar su potencial aplicación en sistemas automáticos de identificación de objetos aéreos.

1.3 Estructura

Tras establecer el contexto del problema y definir los objetivos que orientan este trabajo, a continuación, se presenta la estructura que sigue la memoria, con el fin de organizar de manera clara los contenidos desarrollados y los resultados obtenidos. El documento se divide en cinco capítulos principales.

• Capítulo 1: Introducción y objetivos. Capítulo actual, en el que se presenta el contexto general del trabajo, destacando la importancia de la detección e identificación de drones en el espacio aéreo y la motivación que impulsa el desarrollo de este estudio. Asimismo, se definen los objetivos principales del proyecto y se describe la organización del documento.

• Capítulo 2: Estado del arte. En este apartado se realiza una revisión de los conceptos teóricos y de las tecnologías relacionadas con la detección de drones y la visión artificial. Se analizan las limitaciones de los métodos tradicionales de detección, la evolución de las técnicas de visión por computador y el papel de las redes neuronales convolucionales en tareas de clasificación y detección de objetos.

• Capítulo 3: Desarrollo del modelo. En este capítulo se describen las metodologías empleadas para el desarrollo del sistema de clasificación propuesto. Se aborda la preparación del *dataset*, las transformaciones aplicadas a los datos, el diseño de la arquitectura de la red neuronal convolucional, así como la configuración del proceso de entrenamiento.

• Capítulo 4: Resultados y evaluación. Este capítulo presenta los resultados obtenidos tras el entrenamiento del modelo y su evaluación sobre el conjunto de prueba y sobre imágenes propias. Se analizan distintas métricas de clasificación y se estudia el comportamiento del sistema mediante herramientas de interpretación visual.

• Capítulo 5: Conclusiones y líneas futuras. En el último capítulo se recogen las conclusiones principales del trabajo, analizando el alcance y las limitaciones del modelo desarrollado. Asimismo, se plantean posibles líneas de investigación futura orientadas a mejorar el sistema y ampliar sus aplicaciones.

Finalmente, se incluye la bibliografía utilizada, así como los anexos necesarios para complementar la información presentada en el documento.

2 ESTADO DEL ARTE

2.1 Contextualización

El progresivo desarrollo tecnológico de los vehículos aéreos no tripulados, comúnmente conocidos como drones, ha transformado de manera significativa el panorama aeronáutico tanto en el ámbito civil como en el militar. Su creciente accesibilidad, versatilidad y reducción de costes han favorecido una expansión masiva de estos sistemas, generando nuevas oportunidades, pero también importantes desafíos en materia de seguridad y control del espacio aéreo.

En este contexto, la detección e identificación fiable de drones se ha convertido en una necesidad estratégica, especialmente en entornos sensibles como infraestructuras críticas, espacios aéreos restringidos, eventos multitudinarios o escenarios operativos militares. Uno de los principales retos asociados a esta problemática radica en la dificultad de distinguir estos dispositivos de otros elementos presentes en el entorno aéreo, como las aves, cuya apariencia y comportamiento en vuelo pueden resultar similares bajo determinadas condiciones.

El presente capítulo presenta el marco teórico que fundamenta el desarrollo de este trabajo. Para ello, se analiza en primer lugar la problemática actual asociada a la identificación de drones y las limitaciones de los métodos tradicionales de detección. Posteriormente, se aborda la evolución de la visión artificial, desde los enfoques clásicos hasta los modelos basados en aprendizaje profundo, prestando especial atención a las redes neuronales convolucionales y a los sistemas modernos de detección en tiempo real. Finalmente, se justifica el posicionamiento del presente trabajo dentro del estado actual de la investigación.

2.2 Problemática de la identificación de drones

2.2.1 Aparición masiva de drones comerciales

Los vehículos aéreos no tripulados tuvieron su origen en el ámbito militar, donde fueron empleados principalmente en tareas de reconocimiento, vigilancia y apoyo operativo. En este contexto, los UAV comenzaron a utilizarse como una solución eficaz para la realización de misiones consideradas "*dull, dirty and dangerous"*, es decir, tareas repetitivas, realizadas en entornos hostiles o que implican un elevado riesgo para el personal humano [3]. Con el tiempo, el desarrollo tecnológico de estos sistemas permitió ampliar progresivamente su campo de aplicación más allá del ámbito militar.

La reducción de costes asociada a la reducción del tamaño de sensores, al abaratamiento de los sistemas de navegación y al desarrollo de baterías más eficientes ha permitido que los drones comerciales sean accesibles a un público cada vez más amplio. Como consecuencia, estos dispositivos han dejado de

ser herramientas exclusivas de entornos especializados para convertirse en plataformas de uso habitual en múltiples aplicaciones [3].

Paralelamente, la mejora de la estabilidad de vuelo y la incorporación de sistemas de control asistido han simplificado de manera significativa su manejo, reduciendo la necesidad de conocimientos técnicos avanzados por parte del usuario. Asimismo, la integración de cámaras de alta resolución y sistemas de transmisión en tiempo real ha ampliado su campo de aplicación en áreas como la inspección de infraestructuras, la vigilancia y el ocio, además de la industria cinematográfica y audiovisual. En la Figura 2-1 se muestran algunas de las aplicaciones de uso más destacadas de drones.



Figura 2-1 Diagrama de campos de aplicación de drones en el ámbito profesional [4]

Este proceso de popularización ha provocado un aumento notable del número de drones operando en el espacio aéreo a baja altura, lo que ha generado nuevos retos en términos de seguridad y control. En este contexto, la detección e identificación fiable de estos sistemas se ha convertido en un problema técnico de creciente relevancia, especialmente en entornos sensibles y áreas con restricciones de vuelo [5].

2.2.2 Riesgos en entornos civiles y militares

La etapa actual de desarrollo de los drones, conocida como la segunda ola drónica, presenta un desafío significativo en el ámbito de la seguridad, debido principalmente a una combinación de varios factores tecnológicos, ya enumerados anteriormente. El principal, se debe al pequeño tamaño de estos sistemas, factor que dificulta considerablemente las labores de detección [6].

En entornos civiles, el uso de drones presenta diversos riesgos relacionados con la seguridad pública y la protección de infraestructuras, especialmente en espacios compartidos con otros elementos aéreos, como los aeropuertos. En este contexto, los planes de seguridad de eventos masivos se han visto obligados a incorporar medidas específicas frente a posibles amenazas asociadas al uso indebido de drones. Asimismo, los centros urbanos de grandes ciudades se encuentran expuestos a este tipo de riesgos, poniendo de manifiesto el impacto potencial que puede tener una actuación maliciosa mediante estos dispositivos. Con el fin de prevenir este tipo de situaciones, surge la necesidad de disponer de sistemas capaces de diferenciar de forma fiable los drones de otros elementos no maliciosos presentes en el entorno, como las aves [7].

En entornos militares, los drones son empleados habitualmente en tareas de reconocimiento, vigilancia y, como fin último ataque, lo que incrementa la complejidad de la identificación de amenazas en escenarios operativos. La necesidad de una identificación rápida y fiable se ve dificultada por la

presencia de múltiples elementos en el entorno, así como por condiciones adversas y limitaciones temporales propias de este tipo de contextos de alta intensidad. En estas situaciones, la aparición de falsos positivos, como la confusión entre drones y aves, puede tener un impacto directo en la toma de decisiones operativas, comprometiendo tanto la eficacia de la misión como la seguridad de las fuerzas desplegadas.

Una identificación incorrecta de drones puede tener consecuencias relevantes tanto en entornos civiles como militares. En primer lugar, la generación de falsas alarmas conlleva un elevado coste económico y operativo, al requerir la activación de procedimientos de seguridad y la interrupción de actividades normales. Asimismo, una clasificación errónea puede dar lugar a respuestas desproporcionadas o, por el contrario, tardías, reduciendo la eficacia de los sistemas de protección. En el ámbito civil, estas situaciones pueden derivar en riesgos legales y de seguridad, especialmente en espacios públicos o infraestructuras críticas, en las que se generaría además un revuelo social especialmente perjudicial. En el contexto militar, una identificación imprecisa puede acarrear consecuencias estratégicas, exponiendo a la fuerza a situaciones de alta tensión y generando además probables conflictos diplomáticos, si no los hubiera ya.

En conflictos recientes, como el actual en Oriente Medio o la guerra entre Rusia y Ucrania iniciada en 2022, los vehículos aéreos no tripulados de pequeño tamaño han desempeñado un papel destacado en operaciones de reconocimiento y apoyo táctico. Diversos estudios señalan que estos sistemas multirrotor se han integrado de forma sistemática en las dinámicas operativas del campo de batalla contemporáneo [8]. A esta tendencia se suma el creciente uso de municiones merodeadoras, o “drones kamikaze” como el mostrado en la Figura 2-2 . Estos son sistemas no tripulados capaces de permanecer en vuelo hasta localizar un objetivo y atacarlo directamente.



Figura 2-2 Dron kamikaze Shahed 136 iraní [9]

2.2.3 Dificultad de distinguir drones de aves con sensores tradicionales

A pesar de la amplia variedad de sensores empleados actualmente para la detección de objetos en el entorno aéreo, la diferenciación entre drones y aves continúa representando un desafío significativo. Ambos comparten características similares en cuanto a tamaño, comportamiento en vuelo y firma observable, lo que dificulta su discriminación fiable, especialmente en escenarios complejos o a larga distancia. Esto evidencia que los métodos tradicionales de detección presentan dificultades para distinguir de forma fiable entre elementos con características visuales y dinámicas similares.

Muchos de los sensores tradicionales y simples detectan la presencia de algún objeto, con el fin único de alertar en caso de que haya contacto positivo, pero sin la capacidad de clasificar la naturaleza de este. La similitud entre drones y aves en determinadas situaciones provoca que la detección se base

únicamente en la presencia de un elemento en el espacio aéreo, sin disponer de criterios claros para su clasificación. Como consecuencia, los sistemas de vigilancia pueden generar alertas ambiguas, lo que reduce su fiabilidad y dificulta la toma de decisiones, especialmente en escenarios donde se requiere una respuesta rápida y precisa.

La presencia de condiciones externas que dificulten la observación es amplia y difícilmente evitable. Factores como la meteorología adversa pueden reducir la visibilidad o la calidad de la imagen, mientras que la instalación de sensores en plataformas móviles, como buques, puede introducir inestabilidad adicional. Asimismo, la distancia a la que se encuentra el objetivo respecto al sensor afecta directamente a la capacidad de clasificación, provocando que drones y aves presenten formas y tamaños aparentes muy similares. Estas circunstancias dificultan notablemente la diferenciación entre ambos, llegando incluso a resultar compleja para el propio ojo humano, tal y como se observa en la Figura 2-3 .



Figura 2-3 Dron y pájaro en misma imagen (fuente: propia)

2.3 Otros medios de detección y clasificación

A lo largo del tiempo se han empleado distintos métodos tradicionales para la detección de drones en el entorno aéreo, basados en tecnologías ya consolidadas en ámbitos civiles y militares. Estos sistemas permiten detectar la presencia de objetos en el espacio aéreo y, en determinados casos, aportar información adicional sobre su comportamiento. No obstante, presentan limitaciones significativas cuando se requiere una identificación precisa del objeto detectado, especialmente en escenarios complejos donde coexisten múltiples elementos [10].

Los sistemas de radar han sido ampliamente utilizados para la detección de objetos aéreos debido a su capacidad para operar a largas distancias y en condiciones meteorológicas adversas. Estos sensores permiten detectar la presencia de un objeto en función de la señal reflejada, proporcionando información sobre su posición y velocidad. Todo esto cobra el máximo de sus sentidos en el control de aeronaves de gran porte en cortas y medias distancias. Sin embargo, los drones de pequeño tamaño presentan una sección radar reducida, lo que dificulta su detección fiable. Además, el radar no permite distinguir directamente entre drones y otros objetos con características similares, como aves, generando posibles falsas alarmas y limitando su capacidad de identificación.

La detección mediante radiofrecuencia se basa en la identificación de las señales emitidas por los sistemas de comunicación entre el dron y su operador. Este método resulta eficaz para detectar drones que se encuentren en funcionamiento activo y utilizando enlaces de control conocidos. No obstante, su

eficacia depende de que el dron emita señales detectables, lo que no siempre ocurre, especialmente en sistemas autónomos o configurados para operar en silencio radioeléctrico, cada vez más frecuentes.

Los sistemas acústicos detectan drones a partir del sonido generado por sus motores y hélices. Este enfoque puede resultar útil en entornos cerrados, cercanos y relativamente silenciosos, donde la firma acústica del dron es claramente identificable. Sin embargo, su rendimiento se ve afectado por el ruido ambiental, las condiciones meteorológicas y la distancia al objetivo.

Los sistemas basados en tecnología LIDAR (*Light Detection and Ranging*) constituyen otra alternativa empleada en la detección de objetos aéreos. Estos sistemas utilizan pulsos láser para medir distancias y generar representaciones tridimensionales del entorno, permitiendo detectar la presencia y posición de objetos con alta precisión espacial. No obstante, su rendimiento puede verse condicionado por factores como las condiciones atmosféricas, el alcance efectivo del sensor y la dificultad para discriminar entre objetos de tamaño reducido con características geométricas similares. Estudios recientes han explorado su aplicación en la detección de drones en entornos reales, poniendo de manifiesto tanto su potencial como sus limitaciones operativas [11].

Algunos sistemas modernos de detección han incorporado técnicas avanzadas para mejorar la discriminación entre drones y aves. Un ejemplo de ello son ciertos radares de vigilancia aérea que emplean análisis de firmas micro-Doppler para identificar patrones característicos de movimiento. Mientras que las aves generan variaciones asociadas al batido de sus alas, los drones producen firmas específicas derivadas de la rotación de sus hélices. El análisis de estas diferencias mediante técnicas de procesamiento de señal y de inteligencia artificial permite mejorar la clasificación de los objetos detectados en el espacio aéreo [12].

Asimismo, existen sistemas comerciales de detección de drones que combinan diferentes tecnologías con el objetivo de mejorar la fiabilidad de la identificación. Un ejemplo de ello es el sistema AARTOS desarrollado por la empresa alemana Aaronia AG, que integra diversos sensores como detección por radiofrecuencia, radar y cámaras ópticas o térmicas dentro de una arquitectura de fusión de sensores (Véase en la Figura 2-4). Este enfoque permite detectar, localizar y seguir tanto el dron como su operador mediante el análisis de sus emisiones electromagnéticas y otros datos del entorno. La combinación de múltiples fuentes de información permite mejorar la clasificación de los objetos detectados y reducir la probabilidad de falsas alarmas [13].



Figura 2-4 Sistema AARTOS de detección de drones basado en múltiples sensores [13]

2.4 Visión Artificial aplicada a la detección y clasificación de objetos

2.4.1 Definición de Visión Artificial

La visión artificial, también conocida como visión computacional o "*computer vision*", se define como la tecnología que permite a las "máquinas" ver, entender y extraer información relevante de las imágenes para resolver tareas específicas. Otras fuentes la definen como la tecnología que utiliza algoritmos para permitir que las computadoras extraigan descripciones significativas a partir de imágenes o videos [14]. Su objetivo principal es procesar y analizar capturas del mundo real de manera que puedan ser comprendidas y tratadas por un ordenador. El concepto surgió en la década de los sesenta del siglo pasado y ha evolucionado hasta convertirse en uno de los mejores métodos actuales para la identificación de imágenes y como una de las áreas de trabajo más interesantes dentro de la inteligencia artificial. El desarrollo de esta tecnología se ve favorecido por los avances en capacidad computacional, el procesamiento de grandes volúmenes de datos y la aparición de modelos basados en aprendizaje profundo (*Deep Learning*).

El elemento básico de la visión artificial es el píxel. En términos generales, la visión computacional consiste en aplicar una serie de operaciones matemáticas sobre los píxeles, denominadas filtros, mediante el uso de máscaras para lograr un objetivo determinado. El flujo de trabajo típico se puede observar en la Figura 2-5 y sigue una secuencia lógica que va desde la adquisición de la imagen -mediante sensores digitales o cámaras- y su procesamiento previo hasta la extracción de datos y la posterior toma de decisiones basada en esa información [15].

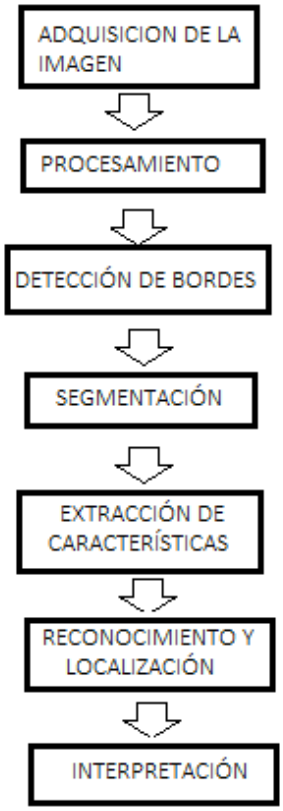


Figura 2-5 Flujo de trabajo típico en un sistema de visión artificial, desde la adquisición de la imagen hasta la interpretación de la información [16]

Esta tecnología tiene aplicaciones muy extendidas hoy en día, que abarcan desde la vigilancia y seguridad, hasta campos más complejos y minuciosos como la medicina -con la interpretación de TACs

y búsqueda de células [17]- o incluso el diseño de interfaces interactivas mediante el reconocimiento de gestos.

De forma análoga al sistema de visión humano, la visión artificial no se limita a la captación de información mediante sensores, sino que un sistema de procesamiento es capaz de interpretar el significado de dicha información para interactuar con el entorno. Pese a ser dos procesos comparables, estos sistemas ven más, mejor, con mayor rapidez y detectan e identifican de una manera diferente a los humanos. Además, es previsible afirmar que esta brecha se amplíe ya que actualmente se podría considerar estamos viviendo una fase inicial del desarrollo de algoritmos, sensores y sistemas autónomos de Inteligencia Artificial [18].

2.4.2 Detección y clasificación de objetos

La detección y clasificación de objetos ocupan un lugar central dentro de las tareas fundamentales de la visión artificial. Su objetivo es permitir que un sistema sea capaz de identificar la presencia de determinados objetos en una imagen o secuencia de vídeo y, en función del objetivo, asignarlos a una o varias categorías predefinidas. Estas tareas resultan esenciales en numerosas aplicaciones, como la vigilancia, la seguridad, la robótica o la inspección automática.

La clasificación de objetos se centra en determinar a qué clase pertenece una imagen o una región de la imagen, asumiendo generalmente que el objeto de interés está presente en la escena. En este caso, el sistema analiza la información visual y asigna una etiqueta que representa la categoría del objeto [14]. Este enfoque ha sido ampliamente utilizado en problemas donde el objetivo principal es distinguir entre diferentes tipos de objetos sin necesidad de conocer su localización exacta dentro de la imagen.

Por su parte, la detección de objetos amplía el problema de clasificación al incorporar información espacial, ya que no solo identifica la clase del objeto, sino que también determina su posición dentro de la imagen mediante regiones delimitadoras [19]. Este tipo de técnicas permite localizar simultáneamente múltiples objetos en una misma escena, lo que resulta especialmente útil en sistemas de vigilancia y análisis de vídeo.

Históricamente, los sistemas de visión artificial se basaban en métodos clásicos, que combinaban la extracción manual de características con clasificadores tradicionales [15]. Sin embargo, estos métodos presentaban limitaciones significativas en escenarios complejos, especialmente cuando los objetos mostraban una elevada variabilidad o aparecían sobre fondos complejos que dificultaban el correcto funcionamiento del proceso. Como consecuencia, su rendimiento se veía reducido en aplicaciones reales de detección y clasificación de objetos.

En las últimas décadas, el desarrollo de técnicas avanzadas de aprendizaje automático ha permitido abordar estos problemas con mayor eficacia, mejorando la capacidad de los sistemas para identificar y diferenciar objetos en condiciones variables. En particular, el análisis automático de imágenes mediante enfoques basados en *Deep Learning* ha supuesto un avance significativo en la detección y clasificación de objetos, sentando las bases de los métodos actuales empleados en visión artificial [20].

2.4.3 Aplicaciones en vigilancia y seguridad

Tal y como se ha introducido en apartados anteriores, la visión artificial desempeña un papel fundamental en los sistemas modernos de vigilancia y seguridad, donde se emplea para el análisis automático de imágenes y secuencias de vídeo captadas por cámaras. Estos sistemas permiten monitorizar de forma continua grandes áreas, reduciendo la necesidad de supervisión humana constante y mejorando la capacidad de respuesta ante posibles incidentes.

En este contexto, las técnicas de detección y clasificación de objetos facilitan la identificación automática de elementos de interés dentro de la escena, como personas, vehículos u otros objetos

relevantes. La automatización de estos procesos resulta especialmente útil en escenarios con grandes volúmenes de datos visuales, donde el análisis manual sería inviable o poco eficiente. Un ejemplo innovador de sistema de detección automática fue el detector de caras propuesto por Paul Viola y Michael Jones, que demostró la viabilidad de realizar detección de objetos en tiempo real mediante el uso de cascadas de clasificadores [21].

Las aplicaciones en vigilancia y seguridad abarcan desde la protección de infraestructuras críticas y espacios públicos hasta el control de accesos y la supervisión de áreas restringidas. En estos entornos, la capacidad de detectar y clasificar correctamente los objetos presentes en la escena constituye un factor clave para la toma de decisiones en tiempo real y la prevención de riesgos, lo que ha impulsado el desarrollo de sistemas cada vez más precisos basados en visión artificial. Estas necesidades, han impulsado el desarrollo de sistemas de vigilancia basados en visión artificial capaces de analizar de forma automática la información visual capturada por cámaras y sensores [22].

El origen de muchas aplicaciones de visión artificial se encuentra en el contexto militar, impulsado por la necesidad de automatizar tareas de vigilancia y reconocimiento en entornos estratégicos. La capacidad de identificar de forma automática los objetos presentes en una escena se convirtió en un elemento clave para la toma de decisiones operativas.

2.4.4 Evolución desde métodos clásicos a Deep Learning

Durante las primeras etapas del desarrollo de la visión artificial, los sistemas de detección y clasificación de objetos se basaban principalmente en la extracción manual de características y el uso de clasificadores tradicionales. En este enfoque, el rendimiento del sistema dependía en gran medida del diseño de descriptores específicos capaces de capturar información relevante sobre bordes, texturas o formas [14], así como de la correcta selección del clasificador empleado.

Entre los descriptores más utilizados en estos sistemas se encontraban aquellos basados en gradientes, texturas o histogramas, que posteriormente eran procesados mediante algoritmos como las máquinas de vectores soporte (SVM), los clasificadores *k-nearest neighbours* (k-NN) o los árboles de decisión. Aunque estos métodos alcanzaron resultados satisfactorios en escenarios controlados, su eficacia se veía limitada cuando se enfrentaban a condiciones reales caracterizadas por variaciones de iluminación, cambios de escala, o fondos complejos.

Estas limitaciones evidenciaron la necesidad de desarrollar enfoques capaces de aprender automáticamente representaciones a partir de los datos, reduciendo la dependencia del diseño manual y del conocimiento experto. En este contexto, el aprendizaje profundo (*Deep Learning*) ha supuesto un avance significativo en la visión artificial. A diferencia de los métodos clásicos, los modelos basados en aprendizaje profundo aprenden directamente a reconocer y clasificar los objetos a partir de los datos. Este enfoque permite aprender representaciones jerárquicas de los datos, desde patrones visuales simples hasta estructuras más complejas, directamente a partir de grandes volúmenes de imágenes etiquetadas [20].

El impacto del aprendizaje profundo ha sido especialmente significativo en tareas de detección y clasificación de objetos, donde ha permitido superar de forma significativa el rendimiento de los métodos tradicionales. Este avance ha sido posible en gran medida gracias al aumento de la capacidad computacional y a la disponibilidad de conjuntos de datos de gran escala [23]. Como consecuencia, estos enfoques se han consolidado como la base de los sistemas modernos de visión artificial, sentando las bases de las arquitecturas actuales empleadas en aplicaciones de vigilancia, seguridad y análisis visual avanzado.

2.5 Redes Neuronales Convolucionales (CNN)

2.5.1 Concepto de Redes Neuronales Convolucionales (CNN)

Las redes neuronales convolucionales (CNN) constituyen una arquitectura específica del *Deep Learning* ampliamente utilizada en el procesamiento de datos con estructura de rejilla, como las imágenes. En la actualidad, representan el enfoque de referencia en tareas de visión artificial, especialmente en problemas de detección y clasificación de objetos [20]. Estas redes permiten procesar información visual de forma jerárquica, aprendiendo representaciones progresivamente más complejas a partir de los datos de entrada, lo que facilita la interpretación automática del contenido visual presente en una imagen.

Desde un punto de vista técnico, una CNN es un modelo de aprendizaje supervisado que emplea la operación de convolución como elemento fundamental en sus capas iniciales, en lugar de la multiplicación matricial general utilizada en redes neuronales densas. Este planteamiento permite explotar interacciones locales, compartir parámetros y preservar la estructura espacial de la imagen, propiedades especialmente adecuadas para el análisis visual [23].

Desde el punto de vista estructural, una red neuronal convolucional está formada por varias capas que trabajan de manera conjunta para analizar una imagen. En primer lugar, las capas convolucionales se encargan de examinar la imagen de entrada y extraer información relevante a partir de pequeñas regiones, lo que permite detectar patrones visuales básicos como bordes o variaciones de intensidad. Este proceso se lleva a cabo mediante la operación de convolución, que constituye el mecanismo fundamental de estas capas.

La operación de convolución consiste en aplicar un pequeño filtro o matriz de pesos sobre diferentes regiones de la imagen de entrada. Este filtro se desplaza a lo largo de la imagen realizando operaciones matemáticas sobre los píxeles de cada región, generando como resultado un nuevo mapa de características. Dependiendo de los valores del filtro, es posible resaltar determinadas propiedades de la imagen, como bordes, texturas o cambios de intensidad. De esta forma, las capas convolucionales permiten extraer progresivamente características visuales cada vez más complejas a medida que se profundiza en la red.

A continuación, las capas de submuestreo -concepto mundialmente conocido como "*pooling*"- reducen el tamaño de la información generada por las capas anteriores, simplificando las representaciones internas del modelo. Este proceso ayuda a que la red sea menos sensible a pequeños cambios en la posición de los objetos dentro de la imagen y mejora su capacidad de generalización [23].

Finalmente, las capas totalmente conectadas, concepto conocido como *fully connected*, utilizan la información extraída en las etapas previas para llevar a cabo la clasificación final o la toma de decisiones. Estas capas están formadas por una serie de capas de neuronas secuenciales que emplean las características extraídas por las capas convolucionales previas para clasificar las imágenes según las clases objetivo predefinidas. De este modo, la red transforma la información visual inicial en una salida que representa la clase del objeto analizado. La arquitectura explicada anteriormente se ilustra en la Figura 2-6 .

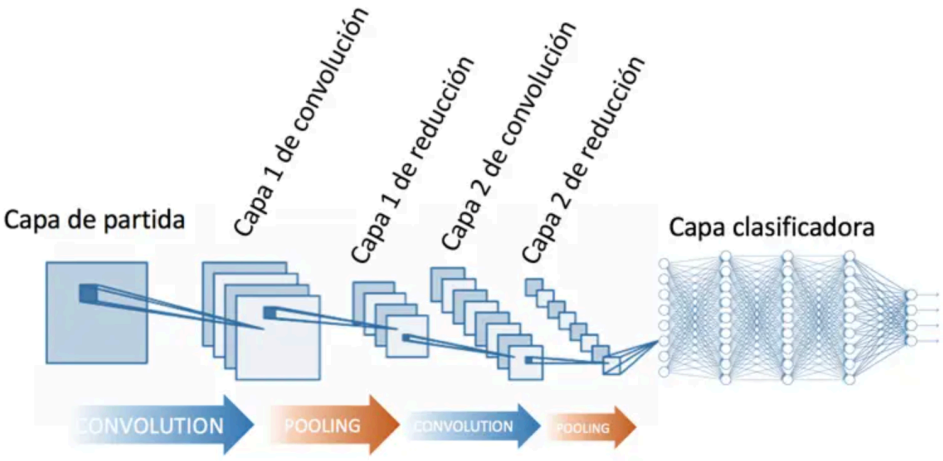


Figura 2-6 Ejemplo de arquitectura de una red neuronal convolucional (CNN) [24]

2.5.2 Redes Neuronales Convolucionales en el procesamiento de imágenes

El elevado rendimiento de las redes neuronales convolucionales en tareas de visión artificial se debe a que su arquitectura respeta la naturaleza espacial de los datos visuales. En una imagen, los píxeles cercanos presentan una fuerte correlación, lo que permite que los filtros convolucionales analicen regiones locales de forma eficiente y sistemática [20]. Este enfoque permite una menor dependencia de la posición del objeto, permitiendo reconocerlo independientemente de su posición dentro de la imagen. Esta propiedad ha demostrado ser clave para alcanzar altos niveles de generalización en grandes conjuntos de datos visuales [25].

Otro aspecto fundamental es el compartido de pesos, mediante el cual un mismo filtro se aplica a distintas regiones de la imagen, en lugar de aprender pesos distintos para cada zona. Este mecanismo reduce de forma significativa el número de parámetros del modelo, disminuyendo el coste computacional y el riesgo de sobreajuste, a la vez que permite detectar patrones visuales de manera consistente en toda la escena [23].

Asimismo, las CNN aprenden características de forma jerárquica, evolucionando desde patrones visuales simples en las primeras capas hasta aquellos de nivel más alto en las capas profundas. Este proceso ha sido ampliamente documentado, mostrando cómo las representaciones internas se vuelven progresivamente más abstractas a medida que aumenta la profundidad de la red [26].

2.5.3 Ventajas frente a métodos tradicionales

Antes de la consolidación de las redes neuronales convolucionales, los sistemas de visión artificial dependían en gran medida de métodos clásicos basados en la extracción manual de características, como SIFT (*Scale-Invariant Feature Transform*) o HOG (*Histogram of Oriented Gradients*), combinados con clasificadores tradicionales. SIFT es un descriptor diseñado para identificar puntos de interés invariantes a cambios de escala y rotación [27], mientras que HOG es un método basado en el análisis de la distribución de gradientes de intensidad para describir la forma de los objetos [28]. Este enfoque requería un elevado conocimiento experto y presentaba limitaciones significativas en escenarios complejos y no controlados.

Las CNN introducen el aprendizaje automático de características (*feature learning*), permitiendo que el propio modelo aprenda durante el entrenamiento qué patrones visuales son relevantes para la tarea considerada [29]. Este planteamiento reduce la participación humana en el proceso y mejora la

adaptabilidad del sistema a problemas específicos sin necesidad de diseñar manualmente descriptores [25].

Además, las redes neuronales convolucionales presentan una eficiencia paramétrica muy superior a la de las redes densas tradicionales, lo que permite entrenar modelos profundos sobre grandes volúmenes de datos. Esta capacidad para manejar grandes cantidades de datos ha posibilitado superar de forma consistente el rendimiento de métodos clásicos como las máquinas de vectores soporte en tareas de detección y clasificación de imágenes [25], consolidando a las CNN como la base de los sistemas modernos de visión artificial [20].

2.6 Modelos actuales de detección en tiempo real

La evolución de los sistemas de detección de objetos ha permitido pasar de enfoques computacionalmente costosos y poco eficientes a modelos capaces de procesar secuencias de vídeo en tiempo real, alcanzando altas tasas de fotogramas por segundo. Este avance ha sido posible gracias al desarrollo de arquitecturas basadas en redes neuronales convolucionales optimizadas para equilibrar precisión y velocidad, entre las que destacan las familias R-CNN**,** SSD y YOLO**.**

2.6.1 La familia R-CNN: detección en dos etapas

El primer gran avance en la detección moderna de objetos se produjo con la familia de modelos R-CNN (*Regions with CNN features*)**,** caracterizados por un enfoque de detección en dos etapas. En estos modelos, el proceso se divide en una fase inicial de generación de regiones de interés y una segunda fase de clasificación.

Las versiones iniciales, como R-CNN y *Fast* R-CNN, utilizaban algoritmos externos para identificar zonas de interés en la imagen que posteriormente eran clasificadas mediante redes convolucionales [30]. Aunque estos métodos supusieron una mejora significativa en precisión, su elevado coste computacional dificultaba su aplicación en escenarios en tiempo real.

Posteriormente, *Faster* R-CNN integró la generación de regiones dentro de la propia red neuronal mediante la *Region Proposal Network* (RPN), mejorando notablemente la eficiencia del sistema. No obstante, a pesar de su alta precisión, este tipo de arquitectura continúa siendo relativamente pesada, lo que limita su uso en escenarios donde se requiere procesamiento en tiempo real sobre hardware con recursos limitados [31].

2.6.2 SSD (Single Shot MultiBox Detector)

Como alternativa a los modelos de dos etapas, surgió SSD, uno de los primeros detectores de una sola etapa (*single-shot detectors*). Este enfoque elimina la generación explícita de regiones de interés y realiza la predicción de las cajas delimitadoras y las clases directamente en una única pasada por la red neuronal.

SSD emplea mapas de características de diferentes escalas para detectar objetos de distintos tamaños, lo que le permite alcanzar un compromiso adecuado entre velocidad y precisión. Gracias a esta arquitectura, SSD resulta considerablemente más rápido que *Faster* R-CNN, aunque puede presentar limitaciones en la detección de objetos muy pequeños o alejados [32].

2.6.3 YOLO (You Only Look Once)

Dentro de los modelos de detección en tiempo real, la familia YOLO (*You Only Look Once*) ha supuesto un cambio, al formular la detección de objetos como un único problema de regresión procesando la imagen completa en una sola pasada de la red neuronal.

YOLO divide la imagen de entrada en una rejilla y asigna a cada celda la responsabilidad de predecir la presencia de objetos, junto con las coordenadas de las cajas delimitadoras y las probabilidades de clase. Este planteamiento reduce de forma drástica la latencia del sistema y permite alcanzar velocidades de procesamiento muy elevadas sin sacrificar excesivamente la precisión [33].

Una de las principales ventajas de YOLO es su capacidad para procesar imágenes a altas tasas de fotogramas por segundo, superando los 45–60 FPS en versiones estándar y alcanzando valores aún mayores en arquitecturas optimizadas, lo que lo hace especialmente adecuado para aplicaciones en tiempo real. Además, al analizar la imagen completa de forma global, el modelo es capaz de comprender mejor el contexto de la escena, reduciendo errores de detección asociados al fondo.

Desde su introducción, YOLO ha evolucionado de forma constante a través de múltiples versiones, incorporando mejoras arquitectónicas y de entrenamiento. Por ejemplo, la introducción de técnicas de optimización en YOLOv4 [34] permitió aumentar la precisión manteniendo un alto rendimiento computacional. Esta evolución continua, consolidada por versiones más recientes como YOLOv8 [35], ha posicionado a YOLO como una de las opciones más utilizadas en aplicaciones prácticas de detección de objetos en tiempo real.

2.6.4 Vision Transformers

En los últimos años han surgido nuevas arquitecturas basadas en Transformers aplicadas al procesamiento de imágenes, conocidas como *Vision Transformers* (ViT). Estos modelos, originalmente desarrollados en el ámbito del procesamiento del lenguaje natural, han demostrado que es posible abordar tareas de visión artificial sin recurrir exclusivamente a convoluciones, utilizando en su lugar mecanismos de atención [36].

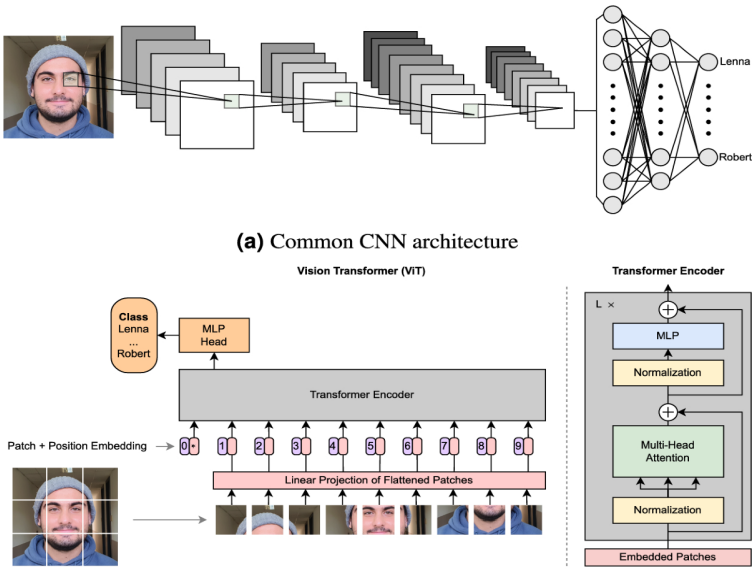
A diferencia de las redes neuronales convolucionales, los *Vision Transformers* dividen la imagen de entrada en pequeños parches (*patches*), que son tratados como secuencias de datos. Cada uno de estos parches se transforma en una representación vectorial (*embedding*) y se procesa mediante mecanismos de autoatención, permitiendo al modelo aprender relaciones globales entre distintas regiones de la imagen de forma directa [36].

Uno de los principales puntos fuertes de los *Vision Transformers* es su capacidad para capturar el contexto global de la imagen desde las primeras capas del modelo, lo que puede resultar ventajoso en tareas donde el contexto global de la imagen es relevante [36]. Mediante técnicas de entrenamiento más eficientes, como las propuestas en *DeiT*, que introdujeron mecanismos de destilación y entrenamiento eficiente [37], estas arquitecturas han logrado resultados competitivos e incluso superiores a los obtenidos con CNN tradicionales en determinadas tareas de clasificación visual."

Sin embargo, los *Vision Transformers* presentan también limitaciones importantes. En general, necesitan conjuntos de datos muy grandes para alcanzar un rendimiento óptimo, ya que no incorporan de forma directa las propiedades estructurales de las CNN, como la localidad espacial y el compartido de pesos [36]. Esto puede dificultar su entrenamiento en problemas donde la disponibilidad de datos etiquetados es limitada o donde se busca una solución eficiente desde el punto de vista computacional. Para mitigar estas carencias, han surgido variantes jerárquicas que reintroducen la localidad mediante ventanas de atención desplazadas, como en el modelo *Swin Transformer* [38].

Por este motivo, aunque los *Vision Transformers* representan una línea de investigación prometedora dentro de la visión artificial, las redes neuronales convolucionales continúan siendo la opción

predominante en aplicaciones prácticas, especialmente en sistemas con restricciones de datos, tiempo de entrenamiento o recursos *hardware*. En este contexto, las CNN han demostrado ser altamente eficaces en una amplia variedad de tareas de visión artificial [20].



(a) Common CNN architecture

(b) Vision Transformer architecture

Figura 2-7 Comparación entre la arquitectura de una red neuronal convolucional (CNN) y un *Vision Transformer* (ViT) [39]

En la Figura 2-7 puede verse que mientras que las redes neuronales convolucionales procesan la imagen mediante filtros locales aplicados jerárquicamente para extraer características espaciales, los *Vision Transformers* dividen la imagen en parches y los tratan como una secuencia, utilizando mecanismos de autoatención para modelar relaciones globales entre distintas regiones desde las primeras capas.

2.7 Uso de *datasets* abiertos y entrenamiento supervisado

El rendimiento de los modelos de aprendizaje profundo aplicados a visión artificial depende de forma directa de la calidad, cantidad y variedad de los datos empleados durante su entrenamiento. En el contexto del aprendizaje supervisado, esta dependencia es especialmente relevante, ya que el modelo ajusta sus parámetros internos a partir de ejemplos previamente etiquetados. En este sentido, el enfoque conocido como *data-centric AI* pone de manifiesto que la mejora del sistema no solo depende de la arquitectura del modelo, sino también de la calidad y gestión de los datos utilizados durante su desarrollo [40].

2.7.1 Importancia de los datasets etiquetados

En el aprendizaje supervisado, el proceso de entrenamiento se basa en conjuntos de datos que contienen tanto la información de entrada, en forma de imágenes, como la respuesta esperada (etiqueta) asociada a cada una de ellas. En tareas de detección de objetos, este etiquetado no se limita a la asignación de una clase, sino que incluye la definición precisa de las coordenadas de las cajas delimitadoras que localizan cada objeto dentro de la imagen.

La disponibilidad de grandes bases de datos anotadas ha sido un factor clave en el avance de la visión artificial moderna. *Datasets* como *LabelMe* [41] o *COCO* [42] han permitido entrenar y evaluar modelos de detección con un alto grado de precisión, favoreciendo el desarrollo de arquitecturas cada vez más eficaces para el análisis automático de imágenes.

La calidad del etiquetado del *dataset* en la fase de entrenamiento resulta determinante. Un etiquetado impreciso introduce ruido durante la fase de optimización, dificultando el aprendizaje del modelo y limitando su capacidad para aprender representaciones visuales fiables. Por tanto, la correcta anotación de los datos constituye un paso esencial para el correcto funcionamiento del sistema.

En este contexto, y atendiendo a los objetivos del presente Trabajo Fin de Grado, se abordará únicamente la tarea de clasificación automática, por lo que se dispondrá de un *dataset* en el que cada imagen estará almacenada en la carpeta correspondiente a su clase, la cual actuará como etiqueta de clasificación.

2.7.2 Limitaciones de los datasets genéricos

Los *datasets* genéricos de gran escala, como *ImageNet* o *MS COCO*, resultan fundamentales en las etapas iniciales de entrenamiento, especialmente cuando se emplean técnicas de aprendizaje por transferencia (*Transfer-learning*), entendiéndose estas como aquellas que, usan un modelo ya entrenado con anterioridad al que se le realizan adaptaciones. Sin embargo, el uso de estos *datasets* presenta limitaciones significativas cuando se aplican a problemas concretos y entornos específicos.

Uno de los principales inconvenientes es el sesgo de dominio, es decir, la diferencia entre los datos de entrenamientos y las condiciones reales de aplicación del sistema [43]. Diferencias en la iluminación, el punto de vista, la resolución o el contexto visual pueden provocar que un modelo entrenado con datos genéricos muestre un rendimiento reducido en situaciones reales. Asimismo, estos conjuntos de datos suelen ofrecer una clasificación generalista de los objetos, lo que puede resultar insuficiente cuando se requiere distinguir entre categorías más específicas o estados particulares de un mismo objeto.

Sin embargo, el principal problema que presentan los *datasets* genéricos es la falta de precisión que ofrecen cuando se aplican a trabajos concretos de análisis de imágenes. El uso de este tipo de conjuntos de datos puede conducir a modelos con un rendimiento limitado, especialmente a la hora de identificar correctamente los objetos que aparecen con menor frecuencia.

No obstante, estos conjuntos de datos continúan siendo una base de gran utilidad en las fases iniciales de desarrollo y entrenamiento de modelos de visión artificial, gracias a su estandarización y a la abundancia de datos disponibles.

2.7.3 Necesidad de datos reales del entorno

Para reducir la diferencia entre los datos de entrenamiento y las condiciones reales de funcionamiento del sistema, resulta necesario incorporar datos obtenidos directamente del entorno de aplicación, con el fin de conseguir un entrenamiento más efectivo. Cuando existe una discrepancia significativa entre la distribución de los datos de entrenamiento y la de los datos reales, se produce el fenómeno ya explicado, conocido como sesgo de dominio o *dataset bias* y puede afectar de forma significativa al comportamiento del modelo si no se aborda adecuadamente [43].

El uso de datos reales permite adaptar el sistema a las condiciones específicas del escenario, incluyendo factores como la configuración de las cámaras, el ángulo de captura, la óptica empleada o las condiciones ambientales propias del entorno. Además, la realización de un ajuste fino (*fine-tuning*) con datos permite adaptar el modelo al nuevo dominio y comprobar que ha aprendido las características visuales relevantes para la aplicación final [44].

La validez de un modelo entrenado se pone de manifiesto cuando es capaz de procesar correctamente datos que mantienen una distribución estadística similar a la del entorno en el que será utilizado. Cuando existe una discrepancia significativa entre ambas distribuciones, el rendimiento del sistema puede verse afectado. Por este motivo, la incorporación de datos capturados en condiciones reales resulta fundamental para mejorar la capacidad de generalización del modelo [23].

Este planteamiento justifica la combinación de *datasets* abiertos con datos propios en la fase experimental del trabajo, permitiendo evaluar el comportamiento del sistema tanto en condiciones estándar como en escenarios representativos del entorno real de aplicación.

En definitiva, queda patente que la calidad del *dataset* desempeña un papel fundamental en el correcto funcionamiento de un sistema de visión artificial. Un etiquetado preciso de los datos resulta esencial para el aprendizaje del modelo, mientras que los *datasets* genéricos, pese a presentar ciertas limitaciones, constituyen una base útil en las fases iniciales de entrenamiento. No obstante, se hace necesaria la incorporación de datos reales del entorno de aplicación, que permitan adaptar el sistema a las condiciones específicas de uso, garantizando un rendimiento adecuado en escenarios concretos.

3 DESARROLLO DEL TFG

3.1 Preparación del *dataset*

3.1.1 Origen del dataset

El presente trabajo se apoya en un *dataset* diseñado específicamente para la tarea de identificación de drones frente a aves, compuesto por imágenes procedentes de diversas fuentes públicas. No se trata de un *dataset* único, sino de un *dataset* compuesto a partir de varios conjuntos de datos diferentes, con el fin de dotar al programa de una gran cantidad de imágenes variadas y crear un nuevo *dataset* adaptado al objetivo concreto del desarrollo de este trabajo.

Las imágenes utilizadas han sido extraídas de plataformas públicas de datos ampliamente utilizadas en el ámbito académico y profesional, como *Kaggle* o *Roboflow*, así como de otros repositorios de datos abiertos. *Kaggle* es una plataforma de referencia en el ámbito de la ciencia de datos, mientras que *Roboflow* está especializada en trabajos y repositorios relacionados con la visión artificial. Los conjuntos de datos empleados han sido publicados por múltiples autores, lo que aporta una mayor diversidad de criterios y escenarios de captura.

Cabe destacar la importancia de que los *datasets* utilizados cuenten con licencia MIT, lo que permite su uso, modificación y reutilización para fines académicos, así como la publicación de resultados obtenidos a partir de sus imágenes, sin ningún tipo de coste ni exigencia adicional más allá de la correspondiente atribución.

El listado de las fuentes empleadas en la elaboración del *dataset* puede consultarse en el Anexo III.

Con el propósito de crear un *dataset* específico para este trabajo, se ha llevado a cabo una selección de imágenes válidas centrada en la tarea final de identificación de drones frente a aves. Para ello, se han combinado imágenes de diferentes especies de aves, con variaciones en forma, tamaño y distancia respecto a la cámara, tanto en vuelo como en tierra, junto con un gran número de imágenes de drones de distintos diseños y dimensiones, capturados a diferentes distancias y sobre fondos y escenarios variados.

3.1.2 Filtrado

Con el objetivo de mejorar la calidad del *dataset* empleado en las diferentes fases del proceso y reducir el ruido presente en los datos antes del entrenamiento, se ha llevado a cabo un filtrado de las imágenes obtenidas de los *datasets* públicos utilizados. Este proceso ha consistido en eliminar aquellas imágenes que se consideró que empeoraban y degradaban el entrenamiento, en lugar de ayudarlo a realizar la tarea planteada. Entre ellas, se encontraron algunas en las que no aparecían ni drones ni aves,

posiblemente incluidas por error en las fuentes originales, así como otras con características consideradas inservibles para el propósito del trabajo.

Pese a haberse realizado un filtrado significativo, eliminando imágenes duplicadas o prácticamente idénticas, se han mantenido numerosas imágenes de características similares cuando estas aportaban variaciones reales (distancia, posición, fondo o perspectiva), con el fin de disponer de un número suficiente de ejemplos para el entrenamiento y favorecer un aprendizaje más robusto.

Asimismo, se han eliminado aquellas imágenes de muy baja calidad (desenfocadas, con escasa resolución o iluminación deficiente) y aquellas en las que el dron o el ave resultaban poco distinguibles incluso para el ojo humano. Se eliminaron fotografías con marcas de agua, se filtraron aquellos fondos inservibles como de interiores o de catálogos con fines publicitarios. Además, se observó como gran parte del *dataset* inicial incluía personas en las imágenes, mandos o incluso *tablets* de control, que hicieron que estas imágenes fuesen recortadas o eliminadas del *dataset*. No obstante, se ha seguido un criterio conservador durante el proceso, evitando descartar casos difíciles pero realistas, con el objetivo de mantener cierta complejidad visual representativa de escenarios reales.

Debido a que numerosas imágenes de drones en el cielo, a diferencia de muchas de aves, incluían el horizonte con tierra, edificios u otros objetos en la parte inferior de la imagen, se recortaron dichas secciones en las imágenes de drones. El objetivo fue evitar la introducción de un sesgo que permitiera al modelo utilizar atajos durante el entrenamiento, como asumir que "si aparece horizonte en la parte inferior, entonces es un dron". Con este ajuste se buscó que el modelo aprendiera principalmente características propias de los objetos y no del fondo de la imagen.

De forma similar, en el *dataset* de pájaros se observó que muchas de las imágenes en las que las aves aparecían con mayor detalle estaban encuadradas en torno al propio objeto, sin incluir demasiado fondo. Por este motivo, en imágenes de drones con características similares también se recortaron los bordes, con el fin de ajustar el encuadre al objeto y mantener una mayor consistencia entre ambos conjuntos de datos.

Tras este proceso de filtrado, se ha conseguido un *dataset* coherente, consistente y adecuado para el entrenamiento supervisado del modelo, mejorando su fiabilidad y reduciendo posibles fuentes de error.

3.1.3 Particionado del dataset

En el desarrollo de modelos de aprendizaje automático es habitual dividir los datos en tres conjuntos diferenciados con el objetivo de que el modelo no se limite a memorizar las imágenes, sino que sea capaz de generalizar ante datos nuevos.

En primer lugar, el conjunto de entrenamiento es el más amplio y es el que se utiliza para que el modelo aprenda los patrones presentes en las imágenes, ajustando progresivamente sus parámetros internos. Es la base sobre la que se construye el aprendizaje. Para tal fin, se ha creado la carpeta *Train*, en la que se ha incluido un 70 % del total de las imágenes, siendo 4057 imágenes de drones y 4808 imágenes de aves (8865 imágenes), constituyendo así el conjunto principal sobre el que el modelo realiza el aprendizaje.

Por otro lado, el conjunto de validación se emplea durante el propio proceso de entrenamiento para comprobar cómo está evolucionando el modelo. Permite ajustar distintos parámetros y detectar posibles problemas como el sobreajuste (*Overfitting*), que ocurre cuando el modelo mejora en el conjunto de entrenamiento, pero empeora en datos que no ha visto directamente. Para ello, se ha definido la carpeta *Validation*, que cuenta con un 15 % del total de las imágenes, con un total de 869 imágenes de drones y 1030 de aves (1899 imágenes).

Finalmente, el conjunto de prueba se mantiene completamente separado hasta el final del proceso. Su función es evaluar de manera objetiva el rendimiento del modelo con datos que no han intervenido

en ninguna fase del entrenamiento, ofreciendo una estimación más realista de su comportamiento en situaciones reales. Para esta prueba final, se ha establecido la carpeta *Test*, que contiene un 15 % de las imágenes totales, distribuidas en 870 imágenes de drones y 1031 de aves, con el fin de obtener una medida final e imparcial del comportamiento del modelo (1901 imágenes).

3.2 Metodología

3.2.1 Papel del dataset en el proceso de trabajo

A la hora de consolidar el *dataset* de entrenamiento, se buscó integrar *datasets* de uso público que permitieran al modelo aprender a identificar drones y pájaros en diferentes condiciones, como distintas distancias o fondos. Para ello, se buscaron y combinaron *datasets* con imágenes que abarcan desde drones y pájaros a gran distancia hasta imágenes en primer plano, con diferentes fondos e iluminaciones. La idea no era únicamente que el modelo aprendiera a distinguir entre pájaros y drones cuando aparecen relativamente lejos, sino también que fuese capaz de clasificar correctamente imágenes claras en las que el objeto aparece cercano y bien definido. Durante el desarrollo del trabajo se prestó especial atención a este aspecto, buscando que el modelo ofreciera resultados fiables y no fallara en situaciones evidentes.

Además, el *dataset* no fue algo estático durante el desarrollo del trabajo. A medida que se realizaban nuevos entrenamientos y se analizaban las imágenes mediante Grad-CAM (que permite observar en qué partes de la imagen se fija el modelo para clasificar), cuando se detectaba alguna carencia o posible sesgo —ya fuese relacionado con el fondo u otro aspecto— no solo se trabajaba en mejorar las transformaciones o la arquitectura de la red. También se dedicó tiempo a revisar y mejorar el propio *dataset*, ya que la calidad de los datos es un aspecto fundamental en los modelos de aprendizaje automático, como se ha explicado en el apartado anterior. En varias ocasiones se revisaron manualmente las imágenes una a una para modificarlas o eliminarlas (una tarea que requirió más de tres días de dedicación), o se buscaron nuevos *datasets* para incorporar imágenes que ayudaran al modelo a aprender mejor las características necesarias.

Como se explicará en otras secciones de la memoria, la capacidad de generalización del modelo y la lucha contra el *overfitting* fueron una preocupación constante durante el desarrollo del trabajo. Por este motivo, además de analizar los resultados obtenidos con el *dataset* de test, también se realizaron pruebas frecuentes con imágenes externas al *dataset* empleado. Estas imágenes, que incluían drones y pájaros seleccionados de forma aleatoria, no se incorporaron al *dataset*, pero se utilizaron como una herramienta útil para detectar visualmente posibles necesidades de mejora, tanto en el propio *dataset* como en la arquitectura del modelo.

3.2.2 Optimización del modelo y transformaciones al dataset

Para llegar a las versiones finales del modelo y a las transformaciones definitivas aplicadas a las imágenes antes del entrenamiento, se siguió un proceso de experimentación basado en los siguientes pasos:

1. Configurar un modelo y las transformaciones aplicadas a las imágenes.
2. Realizar el entrenamiento y evaluar las métricas obtenidas (principalmente *loss* y *accuracy*) en entrenamiento, validación y test.
3. Analizar las imágenes mediante Grad-CAM, incluyendo también imágenes externas al *dataset*.
4. Identificar posibles debilidades y modificar el modelo, las transformaciones y, cuando fue necesario, también el *dataset*.

En este proceso, las métricas no fueron siempre el único criterio para decidir el diseño del modelo y de las transformaciones, especialmente en la fase final del proyecto. En un primer momento sí se

estableció como objetivo alcanzar buenas métricas, concretamente una *accuracy* en test de al menos el 95 %. Sin embargo, al trabajar con los primeros modelos se detectaron problemas de *overfitting*, por lo que se incorporaron transformaciones más agresivas y algunos elementos en la arquitectura destinados a mitigarlo, que se explicarán en el apartado correspondiente. Como consecuencia, las métricas iniciales disminuyeron, ya que el modelo encontraba más dificultades para aprender.

Durante esta fase, las métricas sí se utilizaron como referencia para mejorar el modelo. Para ello se modificó la arquitectura añadiendo más capas convolucionales con distintos tipos de filtros, así como técnicas como ramificaciones, concatenaciones o conexiones de información entre distintas partes del modelo. Tras estos cambios, se volvieron a alcanzar valores elevados de *accuracy*. No obstante, al analizar los resultados se observó que modelos con *accuracy* superior al 93 % y sin evidencias aparentes de *overfitting* en las gráficas de entrenamiento seguían fallando al clasificar algunas imágenes claras de drones o pájaros externas al *dataset*. Además, los mapas Grad-CAM mostraban que en algunos casos el modelo estaba utilizando más información del fondo que del propio objeto.

Por este motivo, en la fase final del proyecto se decidió centrar los esfuerzos en mejorar la capacidad de generalización del modelo y en reducir la influencia del fondo en la predicción, incluso aunque esto implicara empeorar ligeramente las métricas obtenidas durante el entrenamiento.

A continuación (Figura 3-1 Figura 3-2 Figura 3-3), se muestran algunos ejemplos de modelos intermedios en los que todavía no se había conseguido que el modelo centrara su atención principalmente en los objetos de interés.

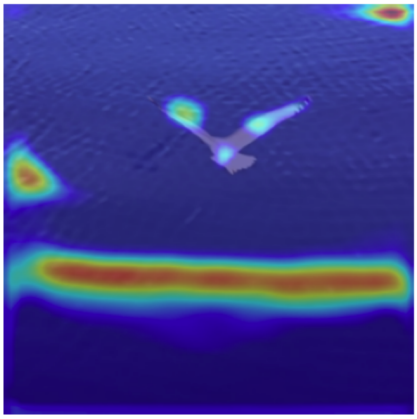


Figura 3-1 Imagen Grad-CAM de ave (fuente: propia)



Figura 3-2 Imagen Grad-CAM de dron 1 (fuente: elaboración propia a partir de imagen tomada de [45])

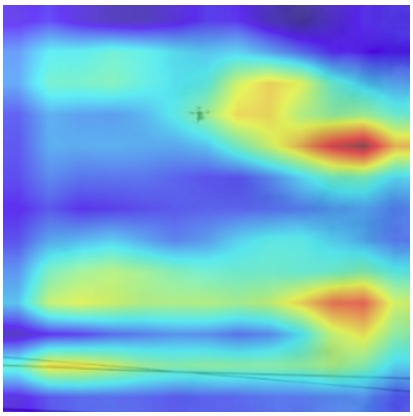


Figura 3-3 Imagen Grad-CAM de dron 2 (fuente: elaboración propia a partir de imagen tomada de [45])

En la Figura 3-4 se puede ver un fallo de identificación claro de un modelo intermedio con una *accuracy* en test del 93.20%:



Figura 3-4 Errores detectados en la metodología (fuente: propia)

En paralelo, se comenzó entrenando con tamaños de imagen de entrada en torno a 320×320 píxeles, que posteriormente se fueron aumentando progresivamente hasta llegar a 448×448. El motivo fue que, cuando los drones o los pájaros aparecían muy pequeños en la imagen, durante el proceso de convolución podían perderse y no ser detectados correctamente por la red. Por ello, se decidió aumentar el tamaño de las imágenes hasta el máximo que permitió la memoria de la GPU disponible, manteniendo un *batch size* de 32. Además, con este mismo objetivo, se incorporó en el modelo un mecanismo que permite conservar información de características extraídas en capas anteriores para reutilizarla en capas finales junto con las últimas obtenidas, tal y como se explicará con más detalle en el apartado de arquitectura de la red.

Tras observar los problemas de generalización y la dependencia del fondo que aparecían en algunos casos, se trabajó en optimizar el *dataset* y se introdujeron cambios en las transformaciones aplicadas a las imágenes, hasta obtener un modelo con resultados que podían considerarse aceptables. No obstante, todavía se observaban dificultades para reducir completamente el efecto del fondo en determinadas situaciones.

Por este motivo, y aprovechando una de las técnicas clásicas de visión artificial como es la detección de bordes, se decidió incorporar un canal adicional de información basado en bordes junto a los tres canales RGB del modelo (RGB + bordes), con el objetivo de comprobar si esto ayudaba a reducir la influencia del fondo en la predicción. Inicialmente se implementó como una transformación aplicada a

las imágenes, pero esto suponía un elevado coste computacional, ya que las transformaciones se realizaban en la CPU. Por ello, finalmente se trasladó este cálculo a la GPU, integrando la extracción de bordes directamente dentro de la estructura de la red convolucional.

En total, el proceso requirió realizar más de 100 entrenamientos, cuya duración fue aumentando desde unos pocos minutos hasta superar las tres horas. Durante este proceso se emplearon modelos con diferentes niveles de complejidad, comenzando con arquitecturas con miles de parámetros y llegando finalmente a modelos con más de cinco millones de parámetros entrenables. Del mismo modo, las transformaciones aplicadas a las imágenes fueron aumentando progresivamente en agresividad, pasando de operaciones sencillas como rotaciones, giros o *zoom*, hasta desplazamientos, modificaciones de color u ocultaciones parciales de la imagen, hasta alcanzar las configuraciones finales descritas en el apartado de arquitectura.

En la fase final del proceso también se realizó una prueba adicional basada en *transfer-learning*. Para ello se utilizó la arquitectura ResNet50 preentrenada en *ImageNet*, con el objetivo de analizar si el uso de características previamente aprendidas podía mejorar el rendimiento del modelo en la tarea de clasificación entre drones y pájaros. Posteriormente se aplicó un proceso de *fine-tuning* sobre las últimas capas de la red para adaptarla al problema concreto abordado en este trabajo, permitiendo comparar este enfoque con el modelo desarrollado para el trabajo.

3.2.3 Optimización del bucle de entrenamiento

Aunque se explicará con más detalle en el apartado correspondiente, en las primeras pruebas se partió de un bucle de entrenamiento básico y se fueron incorporando progresivamente distintos elementos. Entre ellos, se incluyó la posibilidad de guardar el mejor modelo obtenido durante el entrenamiento en lugar del último, un mecanismo de *early stopping* para detener el entrenamiento cuando el modelo dejaba de mejorar, y el almacenamiento de métricas de cada época para poder representar posteriormente las curvas de *loss* y *accuracy*, entre otros.

Aunque la función de pérdida se mantuvo durante todo el proceso, sí se incorporó *label smoothing* con el objetivo de evitar que el modelo memorizara en exceso el conjunto de entrenamiento y así mitigar el *overfitting*. En cuanto al optimizador, aunque en algún entrenamiento puntual se probó el descenso de gradiente estocástico, en la mayoría de los casos se utilizó Adam. Además, se experimentó con dos valores de *learning rate* (0.001 y 0.0005) y se añadió un *scheduler* encargado de reducirlo cuando la mejora del modelo se estancaba durante el entrenamiento.

Por último, también se probaron diferentes valores de *batch size* en función del tamaño de las imágenes utilizadas y de la memoria disponible en la GPU, así como distintos números de épocas totales en los entrenamientos realizados.

3.3 Arquitectura de la red

3.3.1 Introducción

Las arquitecturas de las redes neuronales desarrolladas en este trabajo han sido implementadas *from scratch*[1], siguiendo el paradigma de las redes neuronales convolucionales (CNN). Para su diseño se han tomado como referencia técnicas y configuraciones habituales en modelos consolidados, adaptándolas al problema concreto abordado. No se ha utilizado ninguna arquitectura predefinida ni modelos

[1] entrenamiento del modelo desde cero, sin emplear pesos preentrenados ni arquitecturas completas previamente definidas. Aunque se han tomado como referencia ciertos bloques estructurales consolidados (como módulos tipo *Inception* o conexiones residuales), la arquitectura final ha sido diseñada e implementada específicamente para este trabajo.

preentrenados, sino que se ha construido una propuesta propia desde cero, orientada específicamente a la tarea de clasificación planteada.

Adicionalmente, sí se realizó una prueba basada en *transfer-learning* utilizando la arquitectura ResNet50 preentrenada en *ImageNet*. Posteriormente se aplicó un proceso de *fine-tuning* sobre las últimas capas de la red para adaptarla a la tarea de clasificación entre drones y pájaros.

El objetivo principal del modelo es diferenciar de manera automática entre la presencia de un dron o un ave en una imagen, un problema que exige que la red sea capaz de captar patrones sutiles de forma, contorno y textura. El diseño final es el resultado de un proceso iterativo de prueba y error, en el que se han ajustado progresivamente el número de capas, los tamaños de los *kernels*, las funciones de activación y las técnicas de regularización.

Este enfoque experimental ha permitido desarrollar dos arquitecturas optimizadas para extraer características discriminativas relevantes y mejorar la capacidad de generalización del sistema frente a ambas clases. En las primeras iteraciones, los modelos desarrollados resultaron demasiado simples o presentaban sesgos de fondo significativos. Por ello, se incrementó progresivamente su complejidad y profundidad, con el objetivo de preservar la información capturada en las capas iniciales y mejorar la capacidad de la red para aprender representaciones más significativas.

Asimismo, se incorporaron diversas técnicas destinadas a mitigar el *overfitting*, mejorar el rendimiento frente a objetos de pequeño tamaño y reducir la influencia de correlaciones asociadas al fondo o contexto de las imágenes.

Las dos arquitecturas finales se diferencian en la configuración de los canales de entrada. En la primera se emplearon tres canales por imagen (RGB), mientras que en la segunda se incorporó un cuarto canal adicional obtenido mediante la aplicación de un algoritmo de detección de bordes sobre las imágenes del conjunto de datos (RGB + bordes). Este canal adicional introduce información estructural explícita con el objetivo de hacer al modelo más resistente a posibles sesgos asociados al fondo.

En el Anexo IV, se incluye un esquema de bloques de la arquitectura de la red neuronal convolucional, que permite entender visualmente las distintas operaciones y transformaciones que tienen lugar a lo largo de la red.

3.3.2 Inicialización y entorno de ejecución

Para la construcción y entrenamiento de la red neuronal convolucional se emplea la librería *PyTorch*, *framework* de código abierto ampliamente utilizado en el ámbito del *Deep Learning* y especialmente optimizado para el uso de GPU. Además, se definen las capas y funciones matemáticas necesarias para la arquitectura del modelo y el manejo de tensores[2]. Asimismo, se utilizan librerías auxiliares como *NumPy* para el tratamiento de estructuras numéricas, *Matplotlib* para la visualización de resultados y módulos estándar de Python para la gestión de archivos y el control de la aleatoriedad.

El entrenamiento principal del modelo se realizó utilizando una GPU NVIDIA RTX 3060, con 12 GB de memoria RAM. Esta disponibilidad de memoria permitió trabajar con imágenes de tamaño 448×448 y mantener el *batch size* seleccionado, algo que no habría sido viable con recursos más limitados sin reducir resolución.

El tamaño de entrada de las imágenes se ha fijado en 448×448 píxeles buscando un equilibrio entre nivel de detalle y coste computacional. Aunque en muchos modelos se utilizan tamaños más reducidos, como 224×224 o incluso 32×32 en redes más simples, en este caso se optó por una resolución mayor debido a las características del problema. Dado que el objeto de interés puede aparecer pequeño dentro

[2] En el caso general, un tensor se define como un arreglo multidimensional de números organizado en una rejilla regular con un número variable de ejes [23].

de la imagen, trabajar con más píxeles permite mantener mejor los detalles necesarios para su identificación. Sin embargo, aumentar la resolución también supone un mayor tiempo de entrenamiento y un mayor consumo de memoria, ya que el coste computacional de una capa convolucional crece proporcionalmente al tamaño espacial de los mapas de características [46]. Por ello, se seleccionó este tamaño como punto intermedio entre precisión y eficiencia.

3.3.3 Capa de extracción de bordes

En la arquitectura con cuatro canales de entrada (RGB + bordes), se define una capa de extracción de bordes cuyo objetivo es generar un mapa de contornos a partir de la imagen de entrada, reforzando la información estructural que posteriormente utilizará la red. A diferencia de un preprocesado externo, esta operación se integra directamente dentro del modelo, permitiendo que el cálculo se realice durante el propio entrenamiento y aprovechando la ejecución en GPU. Además, el mapa de bordes obtenido se utilizará como canal adicional de entrada, reduciendo la influencia del fondo en la extracción de características.

En el método *init* se preparan los elementos necesarios para el filtrado. En primer lugar, se construye un *kernel* Gaussiano 3×3, normalizado dividiendo entre 16, cuya función es suavizar la imagen y reducir pequeñas variaciones de textura que podrían generar bordes irrelevantes. Este *kernel* se almacena mediante *register_buffer*, de modo que forma parte del modelo sin ser un parámetro entrenable.

A continuación, se definen los operadores Sobel en dirección horizontal y vertical. Estos filtros permiten estimar las variaciones de intensidad en ambos ejes espaciales, facilitando la detección de contornos [14]. Al igual que el filtro Gaussiano, se almacenan como buffers fijos.

Por último, se define la variable *threshold*, fijada inicialmente en 0,15, que actúa como criterio para eliminar bordes de baja intensidad. Este valor puede ajustarse en función del comportamiento observado durante las pruebas: aumentarlo (por ejemplo, a 0,2) permite eliminar mayor cantidad de ruido de fondo, mientras que reducirlo (por ejemplo, a 0,1) hace que se conserven bordes más débiles, aunque con mayor riesgo de introducir información irrelevante.

En el método forward se desarrolla el flujo completo de procesamiento, inicializado en el método *init*. En primer lugar, la imagen RGB se convierte a escala de grises mediante una combinación lineal ponderada de sus tres canales. Seguidamente, se aplica el suavizado Gaussiano mediante una convolución con *padding*=1, lo que mantiene constantes las dimensiones espaciales y evita la pérdida de información en los bordes.

Sobre la imagen suavizada se aplican los filtros Sobel en ambas direcciones, obteniendo dos mapas de gradiente (grad_x y grad_y). A partir de ambos gradientes se calcula la magnitud del gradiente, combinando las componentes horizontal y vertical para obtener una medida única de la intensidad del borde en cada píxel.

Finalmente, se realiza un umbralado suave restando el valor del umbral y aplicando la función *ReLU*. Matemáticamente, esta función se define como:

$$ReLU(x) = \max(0,x)$$

Esta función introduce no linealidad en el modelo, anulando los valores negativos y manteniendo inalterados los positivos. En el contexto de redes convolucionales, su uso es fundamental, ya que sin funciones de activación no lineales la red se reduciría a una transformación lineal, limitando su capacidad de aprendizaje.

En este caso concreto, la función *ReLU* elimina automáticamente los valores por debajo del umbral y conserva únicamente los contornos más relevantes. Además de su sencillez computacional, *ReLU* ayuda a evitar problemas asociados al gradiente evanescente durante el entrenamiento, facilitando una

actualización más estable de los pesos en redes profundas [47]. El resultado final es un mapa de bordes limpio que se incorporará como información estructural adicional en la arquitectura completa.

3.3.4 Arquitectura principal DeepCNN

A continuación, se describe en detalle la arquitectura del modelo de cuatro canales (RGB + bordes), dado que la arquitectura del modelo de tres canales (RGB) es equivalente, diferenciándose únicamente en la ausencia de la etapa de extracción de bordes y en el uso de tres canales de entrada en lugar de cuatro.

La red incorpora en primer lugar la capa de bordes definida anteriormente, integrando así el mapa de contornos dentro de la arquitectura principal. Durante el paso forward, este mapa se concatena con la imagen original mediante torch.cat, generando una entrada de cuatro canales (3 RGB + 1 de bordes). Todo ello se incluye visualmente en la Figura 3-5 .

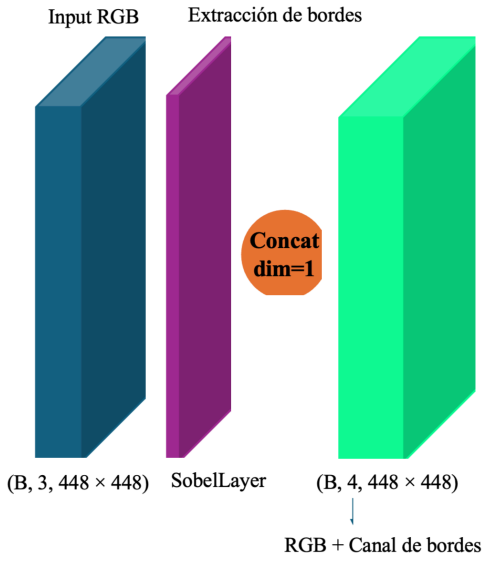


Figura 3-5 Concatenación del canal de bordes con la imagen RGB para generar una entrada de cuatro canales (fuente: propia)

El primer bloque convolucional está formado por dos capas *Conv2d*. La primera (conv1) recibe 4 canales de entrada y genera 32 mapas de características mediante filtros de tamaño 3×3 con *padding*=1, lo que permite conservar el tamaño espacial. La capa convolucional crea 32 filtros distintos y cada uno de esos 32 filtros tiene una profundidad de 4 para coincidir con los 4 canales de entrada. Cuando la operación se ejecuta, la información de los 4 canales se suma y se comprime, dando como resultado un único canal por cada filtro, resultando 32 mapas de características en total.

A continuación, se aplica una capa *BatchNorm2d*, que normaliza las activaciones y estabiliza el entrenamiento. Esta capa recibe los 32 mapas de características obtenidos y estandariza los valores de sus píxeles matemáticamente, evitando que la red se desestabilice si algunos canales producen números excesivamente grandes o minúsculos. Tras la normalización se aplica la función de activación *ReLU*, ya explicada anteriormente, para introducir no linealidad y que la red pueda aprender patrones complejos.

La segunda convolución (*conv2*) vuelve a operar con 32 canales de entrada y salida, manteniendo el tamaño de *kernel* 3×3. Mientras que la primera convolución detecta patrones básicos como bordes o líneas, esta segunda comienza a combinar dicha información para detectar formas más complejas.

Posteriormente, se aplica una operación de *MaxPool2d* (2,2) que reduce las dimensiones espaciales a la mitad, disminuyendo el coste computacional y permitiendo que las siguientes capas trabajen con un campo receptivo mayor. En este paso la red examina pequeños bloques cuadrados (2×2) de cada mapa de características y se queda únicamente con el valor de píxel más alto (la "señal" más fuerte o la característica más activada), descartando el resto. Al aumentar el campo receptivo al encoger el mapa de características a 224×224, un filtro de 3×3 en la siguiente capa convolucional ya no evaluará un área de 3x3 píxeles de la imagen original, sino un área equivalente mucho mayor. Esto permite que la red pase de entender detalles finos (como bordes o texturas) a comprender el "contexto" global de la forma (como un ala o una hélice).

Ambas convoluciones y la última operación explicada se observan en la Figura 3-6 .

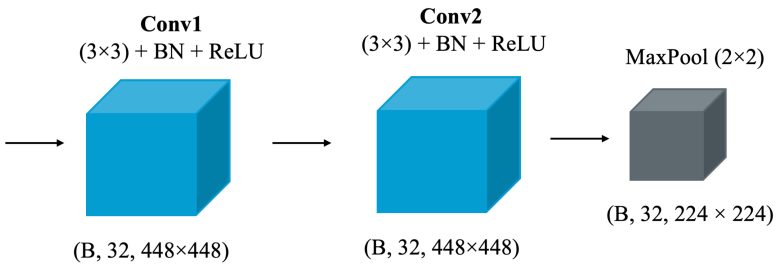


Figura 3-6 Primer bloque convolucional de la red con capas Conv, BatchNorm, ReLU y MaxPooling (fuente: propia)

A continuación, se introducen dos bloques consecutivos de tipo *Inception* [3] [48]. En cada uno de ellos se realizan tres convoluciones en paralelo con distintos tamaños de *kernel* (1×1, 3×3 y 5×5). Esta estructura permite analizar la imagen a distintas escalas de forma simultánea: los *kernels* más pequeños capturan detalles locales, mientras que los mayores detectan patrones más amplios.

En el primer bloque, cada rama genera 32 mapas de características, que posteriormente se concatenan en la dimensión de canales, dando lugar a 96 canales de salida (3 × 32). Tras la concatenación, se aplica una operación de *max pooling* para reducir nuevamente la resolución espacial (Figura 3-7).

[3] El bloque *Inception* es un módulo de red neuronal que aplica simultáneamente filtros de distintos tamaños (1x1, 3x3, 5x5) y una operación de *pooling* sobre la misma entrada, concatenando sus resultados para capturar características a diferentes escalas de forma eficiente [48] .

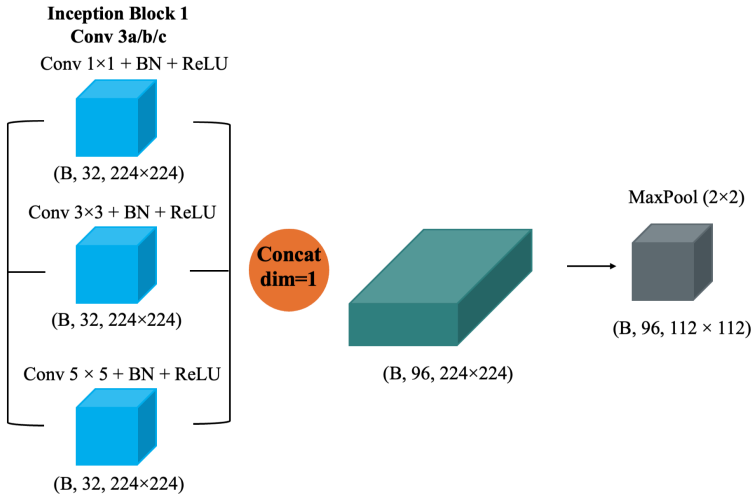


Figura 3-7 Primer bloque tipo *Inception* con convoluciones en paralelo y concatenación de características (fuente: propia)

En el segundo bloque se mantiene la misma filosofía, pero aumentando la profundidad: cada rama genera 64 mapas, partiendo ahora de 96 canales de entrada. Esto incrementa la capacidad representativa del modelo, permitiendo detectar patrones progresivamente más complejos (Figura 3-8).

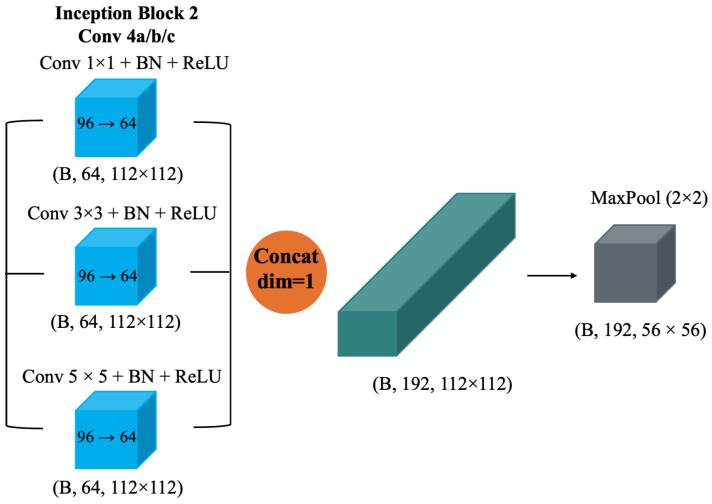


Figura 3-8 Segundo bloque *Inception* con convoluciones en paralelo y concatenación de características (fuente: propia)

Tras los bloques *Inception*, se aplica otro con dos convoluciones adicionales (conv6a y conv6b) que reciben 192 canales de entrada. La primera utiliza un *kernel* 3×3 convencional, mientras que la segunda incorpora dilatación (*dilation*=2). Véase en la Figura 3-9 .

El uso de convolución dilatada permite ampliar el campo receptivo sin incrementar el número de parámetros, ya que el filtro "salta" posiciones dentro de la ventana. De este modo, se combinan patrones locales y más globales, enriqueciendo la representación interna.

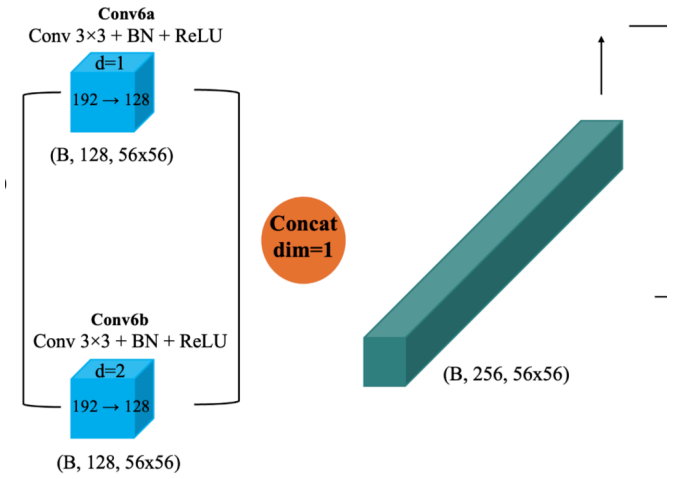


Figura 3-9 Convoluciones paralelas con diferentes dilataciones y concatenación de características (fuente: propia)

En este punto se aplica una capa *AdaptiveAvgPool2d*((1,1)), denominada *gap_mid* y esquemáticamente visible en la Figura 3-10 . Esta operación calcula la media de cada mapa de características, convirtiendo cada uno en un único valor representativo. Así, la información espacial se resume en un vector compacto que recoge qué patrones han sido activados, independientemente de su posición exacta en la imagen.

Además de su función estructural, esta operación actúa como un regularizador para combatir el sobreajuste (*overfitting*). A diferencia del enfoque clásico de únicamente aplanar los tensores espaciales (*Flatten*) para pasarlos por densas capas lineales, que añade millones de parámetros propensos a memorizar el ruido de los datos de entrenamiento, el *Global Average Pooling* no introduce ningún parámetro entrenable adicional [49]. Esto fuerza a la red a aprender mapas de características más robustos y generalizables, reforzando el valor de las características obtenidas y restando importancia a su ubicación espacial, lo que mitiga la tendencia a la memorización y favorece el aprendizaje.

Este paso se realiza para guardar las características obtenidas hasta el momento con el nivel de convolución alcanzado y, posteriormente, concatenarlo con el resultado de las convoluciones posteriores. Esto permite que, si el objeto a identificar (dron/pájaro) desaparece en futuras capas al ser demasiado pequeño, no se pierdan sus características fundamentales. Este procedimiento es similar a lo que hacen arquitecturas como *DenseNet* [50] o las redes de Pirámide de Características (FPN) [51] y pretende incorporar capacidad de memoria de características al modelo.

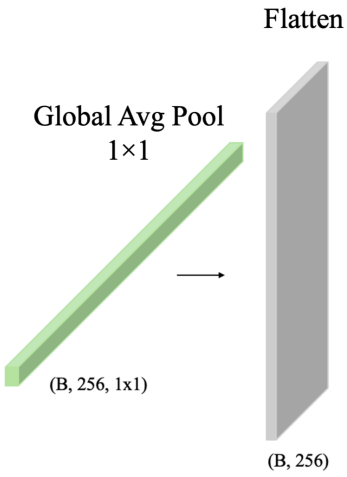


Figura 3-10 Operación de *Global Average Pooling* seguida de *flatten* (fuente: propia)

Tras la extracción de características intermedias mediante *gap_mid* a modo de memoria, la red continúa el proceso con una etapa de reducción denominada conv7_reducer. Esta capa aplica un filtro 3×3 con *stride* = 2, lo que permite reducir nuevamente las dimensiones espaciales a la mitad manteniendo los 256 canales de salida. Como se observa en arquitecturas modernas como las *redes All-Convolutional* [52] los bloques de reducción de ResNet [53] se prefiere el *downsampling* mediante convoluciones. Esto permite que la red aprenda, a través de sus pesos, qué información espacial descartar y qué conservar, superando la rigidez del *MaxPool*, que se queda por defecto con el píxel de mayor valor numérico.

A continuación, la capa conv8 incrementa la profundidad de 256 a 512 canales, aumentando la capacidad representativa del modelo en esta etapa profunda. En paralelo, se implementa una rama de conexión residual (*residual_shortcut*) compuesta por una convolución 1×1. Esta actúa como un adaptador para transformar los 256 canales originales de salida de la capa conv7_reducer en 512, permitiendo que la información original se sume directamente a la salida de conv8. Esta estructura de *skip connection* facilita el flujo del gradiente y mejora la estabilidad del entrenamiento [52]. Finalmente, tras la suma de ambas ramas, se aplica la activación *ReLU* sobre el conjunto. Las dos últimas convoluciones explicadas y la activación *ReLU* se muestran gráficamente en la Figura 3-11 .

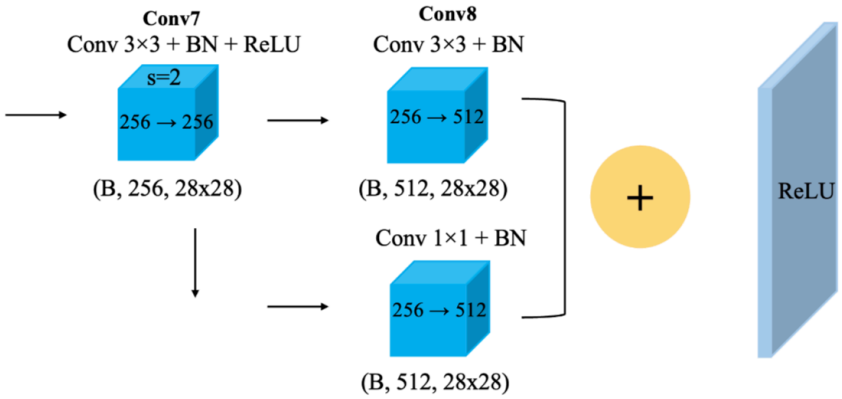


Figura 3-11 Bloque residual con *skip connection* y activación *ReLU* (fuente: propia)

Posteriormente, se aplica una operación de *MaxPool2d(2,2)* (*pool_final*) para reducir nuevamente el tamaño espacial. A continuación, la capa *conv9* mantiene los 512 canales de entrada y salida, refinando la representación final antes de la etapa de clasificación.

Para reducir el riesgo de sobreajuste, se incorpora una capa de *Dropout2d* con probabilidad 0,2 (*spatial_dropout*), que desactiva aleatoriamente mapas completos de características durante el entrenamiento, obligando al modelo a no depender excesivamente de activaciones concretas.

Al omitir canales enteros, se rompe la dependencia espacial entre píxeles adyacentes y se fuerza a la red a no confiar en una única característica (como una textura específica del dron), obligándola a aprender representaciones más diversas, robustas y generalizables.

Finalmente, se aplica una segunda capa de promedio global adaptativo *AdaptiveAvgPool2d*((1,1)), denominada *gap_final*, que reduce cada mapa a un valor único. El tensor resultante se aplana (*flatten*), obteniendo un vector de 512 características que representa la información semántica más abstracta y profunda de la red. Estos últimos bloques han sido incluidos en la Figura 3-12 .

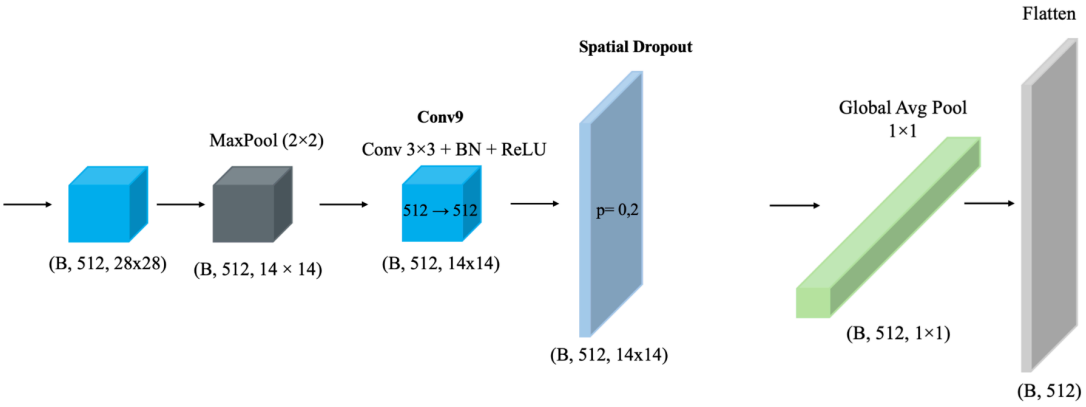


Figura 3-12 Etapas finales de la red: *pooling*, convolución, *dropout* espacial y *Global Average Pooling* (fuente: propia)

Tras la obtención de las características finales mediante *gap_final* y su correspondiente *flatten*, se procede a combinar esta representación con las características intermedias extraídas previamente mediante *gap_mid*. Ambas se concatenan en la dimensión de canales utilizando *torch.cat*, dando lugar a un vector de 768 características (512 finales + 256 intermedias).

Esta concatenación (Figura 3-13) permite integrar información de distintos niveles de abstracción: mientras que las características finales capturan la semántica global, las intermedias conservan detalles estructurales críticos que podrían haberse diluido en las capas más profundas.

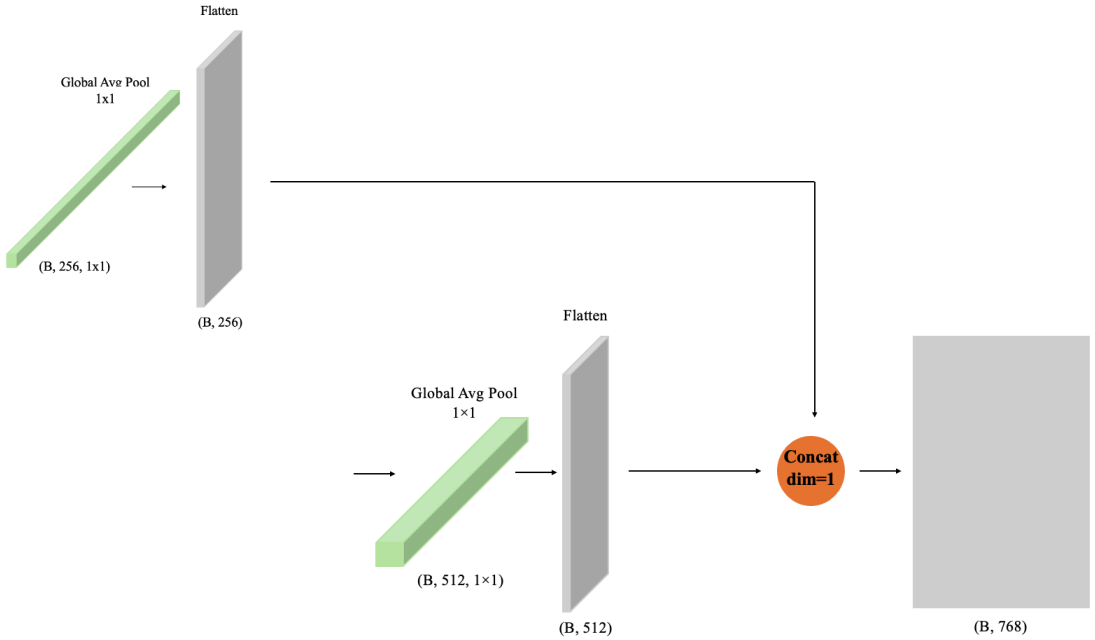


Figura 3-13 Concatenación de los vectores de características antes de la capa clasificadora (fuente: propia)

El vector resultante se introduce en una primera capa totalmente conectada (*fc1*), que reduce la dimensionalidad de 768 a 128 neuronas, seguida de activación *ReLU*. A continuación, se aplica una capa de *Dropout* con probabilidad 0,5 (*dropout1*) para reducir el riesgo de sobreajuste.

Posteriormente, una segunda capa lineal (*fc3*) reduce la representación a 32 neuronas, nuevamente seguida de activación *ReLU* y otra capa de *Dropout* (*dropout3*). Finalmente, la capa *fc4* proyecta la representación en un vector de dimensión 2, correspondiente a las clases del problema (dron o pájaro). Estas últimas operaciones se ilustran en la Figura 3-14 .

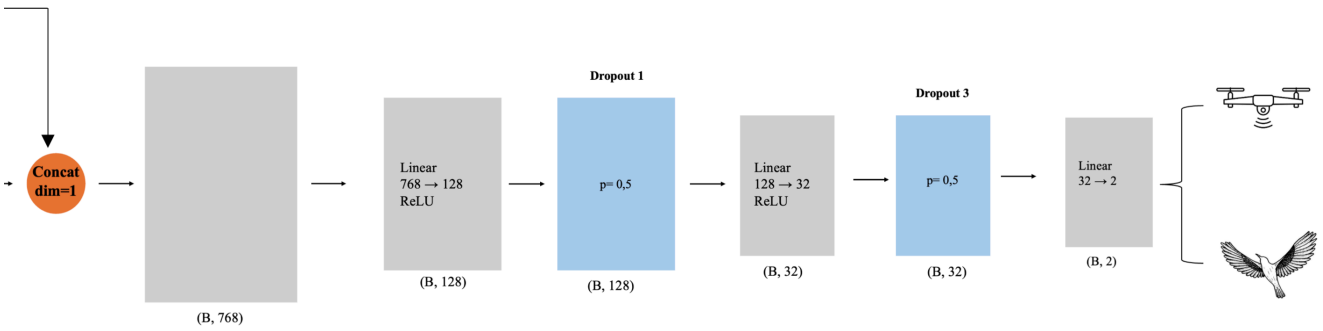


Figura 3-14 Capas finales de clasificación con capas lineales y *dropout* (fuente: propia)

3.3.5 Resumen técnico de parámetros

En el Anexo V se muestran las tablas resumen de ambos modelos, donde se puede ver la estructura completa de la red y las capas que la componen. Como se observa, ambos modelos tienen una arquitectura muy similar, diferenciándose únicamente en la capa inicial, donde en uno de ellos se añade el operador de Sobel para incorporar la información de bordes como un canal adicional de entrada. En cuanto a su tamaño, ambos modelos cuentan con algo más de 5 millones de parámetros entrenables.

3.3.6 Proceso de carga y preparación de datos de entrada

Antes de comenzar el entrenamiento, se fijan las rutas correspondientes a los subconjuntos previamente generados (*train*, *validation* y *test*), garantizando la correcta separación de los datos.

Posteriormente, se establecen dos conjuntos de transformaciones diferenciadas: uno aplicado al conjunto de entrenamiento y otro destinado a validación y prueba. El uso de transformaciones diferenciadas en el conjunto de entrenamiento contribuye a mejorar la capacidad de generalización del modelo, reduciendo el riesgo de sobreajuste (*Overfitting*).

Estas transformaciones permiten adaptar las imágenes al formato de entrada requerido por la red, convertirlas a tensores y normalizarlas antes de su procesamiento. El detalle completo de dichas transformaciones se desarrolla en el apartado específico dedicado a la preparación y aumento del *dataset*.

Finalmente, se crean los objetos *DataLoader*, que permiten cargar las imágenes en lotes (*batch size* = 32) y gestionar su envío eficiente al dispositivo de cómputo. Asimismo, se fija una semilla global (*Seed*) para asegurar que los resultados obtenidos puedan reproducirse en ejecuciones posteriores bajo las mismas condiciones.

3.4 Transformaciones aplicadas a los datos

El presente apartado describe las transformaciones aplicadas al *dataset* con el objetivo de mejorar la capacidad de generalización del modelo y reducir el riesgo de sobreajuste. Para ello, se han definido dos vertientes diferenciadas: una para entrenamiento (con aumento de datos) y otra para validación y test (sin aumentos aleatorios).

En conjunto, estas transformaciones tienen como objetivo principal aumentar artificialmente la variabilidad del conjunto de entrenamiento, generando múltiples versiones de una misma imagen mediante modificaciones geométricas, cromáticas y estructurales. Este incremento controlado de la diversidad introduce mayor dificultad durante el proceso de aprendizaje, obligando al modelo a extraer características más abstractas y robustas. Como consecuencia, se reduce la dependencia de patrones superficiales, como las correlaciones asociadas al fondo o contexto de la imagen, y se mejora la capacidad de generalización frente a imágenes reales no vistas.

Con el fin de garantizar la reproducibilidad de los resultados, se fijó una semilla global (*SEED* = 42) que controla los procesos aleatorios implicados en el entrenamiento y en las transformaciones aplicadas a los datos. Esta configuración afecta a:

- Generadores aleatorios de Python
- *NumPy*
- *PyTorch* (CPU y GPU)
- Configuración determinista de CUDA

De este modo, las transformaciones aleatorias y el proceso de entrenamiento pueden replicarse en ejecuciones posteriores.

Durante el entrenamiento se aplican técnicas de *data augmentation* para generar variaciones artificiales de las imágenes originales, aumentando la diversidad del conjunto sin necesidad de recopilar nuevos datos. Las transformaciones aplicadas son:

- *Pad*

En primer lugar, se aplica un *padding* reflectivo de 64 píxeles alrededor de la imagen original. Esta operación amplía artificialmente los bordes de la imagen replicando de forma reflejada los píxeles existentes, lo que permite que el objeto pueda desplazarse dentro de la imagen sin desaparecer durante

las transformaciones posteriores. De este modo se favorece la generación de recortes y traslaciones más variadas sin perder información relevante del objeto.

- *RandomResizedCrop*

A continuación, se aplica *RandomResizedCrop*, que realiza un recorte aleatorio de la imagen seguido de un redimensionamiento al tamaño de entrada del modelo. En este caso, el área recortada puede variar entre el 50% y el 100% de la imagen original. Esta operación simula diferentes niveles de zoom y obliga al modelo a reconocer el objeto incluso cuando aparece a distintas escalas o en posiciones variables dentro de la imagen, además de reducir el efecto del fondo en el aprendizaje.

- *RandomAffine*

Posteriormente se aplica una transformación afín que introduce traslaciones aleatorias de hasta un 30% tanto en el eje horizontal como vertical. Esta transformación permite simular desplazamientos del objeto dentro del encuadre, evitando que el modelo dependa excesivamente de su posición exacta en la imagen.

- *RandomHorizontalFlip* y *RandomRotation*

En cuanto a las transformaciones geométricas, se aplican operaciones como *RandomHorizontalFlip*, que invierte horizontalmente la imagen con una probabilidad del 50%, y *RandomRotation* (±15°), que introduce pequeñas rotaciones de hasta 15º. Estas modificaciones permiten que el modelo sea menos sensible frente a cambios de orientación, algo especialmente relevante en objetos voladores como drones y aves.

- *ColorJitter*

En cuanto a las transformaciones cromáticas, se aplica *ColorJitter*, que modifica aleatoriamente parámetros como el brillo, el contraste, la saturación y el tono de la imagen. El objetivo es evitar que el modelo dependa excesivamente de la información cromática, especialmente en escenarios con nubes o cielos brillantes. De este modo, se favorece el aprendizaje de características estructurales, como la forma, las alas o las hélices, en lugar de basarse únicamente en el color.

- *RandomGrayscale*

Asimismo, se aplica *RandomGrayscale*, que convierte la imagen a escala de grises con una probabilidad del 10%. Esta transformación refuerza la idea anterior, obligando al modelo a centrarse en la morfología y la estructura del objeto en lugar de depender exclusivamente de la información de color, tanto en el propio objeto como en el fondo de la imagen.

- *ToTensor* y *Normalize*

A continuación, se aplican las transformaciones *ToTensor* y *Normalize*. La primera convierte la imagen en un tensor compatible con *PyTorch*, permitiendo su procesamiento por la red neuronal. Posteriormente, *Normalize* ajusta los canales RGB utilizando las medias y desviaciones estándar del conjunto *ImageNet*. Este proceso contribuye a estabilizar el entrenamiento y facilita la convergencia del modelo.

- *RandomErasing*

Finalmente, se aplica la transformación *RandomErasing*, que elimina aleatoriamente una pequeña región de la imagen con una probabilidad del 30%. Esta técnica incrementa la robustez del modelo frente a oclusiones parciales, elementos como nubes que puedan cubrir parte del objeto o presencia de ruido visual. De este modo, el modelo aprende a identificar el objeto incluso cuando no se encuentra completamente visible. Además, al ocultar ocasionalmente regiones del fondo, se reduce la tendencia del modelo a aprender correlaciones asociadas al contexto de la imagen en lugar de centrarse en las características propias del objeto.

El orden de aplicación de las transformaciones sigue la práctica habitual en visión por computador, aplicando primero modificaciones geométricas y cromáticas sobre la imagen original y, posteriormente, la conversión a tensor y normalización necesarias para su procesamiento por la red neuronal [54].

Tras configurar el *dataset* y las transformaciones, se incluyó un bloque independiente de inspección visual que permite comprobar la correcta carga y etiquetado de las imágenes antes del entrenamiento (Figura 3-15). Se realizó una inspección visual de los lotes de entrenamiento mostrando ocho imágenes por lote, con el fin de evitar la saturación visual. Para su correcta visualización, las imágenes fueron desnormalizadas previamente, recuperando los valores originales de color. Además, se mostró la clase asociada a cada imagen según el etiquetado automático del *dataset*.

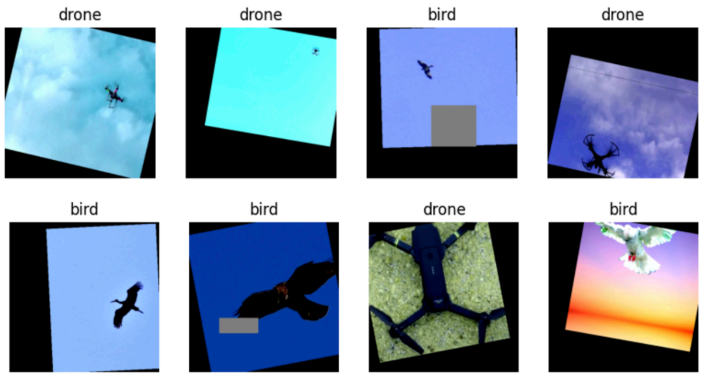


Figura 3-15 Ejemplo de lote de entrenamiento tras la aplicación de transformaciones y desnormalización para su visualización (fuente: propia)

En los conjuntos de validación y prueba no se aplican aumentos aleatorios, ya que el objetivo es evaluar el rendimiento real del modelo en condiciones controladas. En su lugar, se emplean las siguientes transformaciones:

Resize, ajusta el tamaño de la imagen redimensionándola a un tamaño ligeramente superior al utilizado como entrada del modelo.

CenterCrop, que realiza un recorte central al tamaño definido, asegurando que todas las imágenes tengan las mismas dimensiones de entrada y que el contenido principal quede encuadrado de forma consistente.

Posteriormente, se aplican *ToTensor* y *Normalize*, con el fin de convertir la imagen en un tensor compatible con la red y ajustar los canales RGB utilizando las medias y desviaciones estándar del conjunto *ImageNet*.

De este modo, las imágenes utilizadas durante la evaluación mantienen una transformación uniforme y controlada, permitiendo analizar el comportamiento del modelo sobre datos no alterados por aumentos aleatorios y facilitando una interpretación más clara de los resultados obtenidos.

3.5 Configuración del entrenamiento

3.5.1 Optimizador del entrenamiento

El proceso de entrenamiento del modelo se basa en un esquema de optimización supervisada orientado a minimizar la función de pérdida sobre el conjunto de entrenamiento. Para ello, se definen tres componentes fundamentales: la función de pérdida, el algoritmo de actualización de pesos y la estrategia de ajuste dinámico del *learning rate*.

Como función de pérdida se utiliza *CrossEntropyLoss*, estándar en problemas de clasificación. En esta arquitectura, la última capa del modelo devuelve dos valores, correspondientes a las clases dron y ave. Esta función transforma internamente esas salidas en probabilidades y calcula el error comparándolas con la etiqueta real de cada imagen, permitiendo ajustar los pesos del modelo durante el entrenamiento.

Además, se incorpora un término de *label smoothing* con valor 0.1. Esta técnica suaviza ligeramente las etiquetas, evitando que el modelo asigne probabilidad absoluta a una única clase y reduciendo así el riesgo de sobreajuste (*Overfitting*).

Para la actualización de los pesos se emplea el optimizador Adam, aplicado sobre todos los parámetros entrenables del modelo. El *learning rate* inicial se fija en 0.0005, buscando un equilibrio entre velocidad de aprendizaje y estabilidad durante el entrenamiento.

Por último, se incorpora un *scheduler* del tipo *ReduceLROnPlateau*, que reduce el *learning rate* cuando la pérdida deja de mejorar durante varias épocas consecutivas. En este caso, si no hay mejora en cinco épocas, la tasa de aprendizaje se reduce multiplicándola por 0.1, estableciendo además un límite mínimo de 10^{-6}. De este modo, el aprendizaje se ajusta dinámicamente según la evolución real del entrenamiento.

3.5.2 Criterio de finalización del entrenamiento

Se define una clase propia denominada *EarlyStopping*, encargada de supervisar la evolución de la pérdida de validación durante el entrenamiento.

En el método *init* se inicializan los parámetros que regulan su funcionamiento. El parámetro *patience*=7 indica que, si la pérdida de validación no mejora durante siete épocas consecutivas, se activará la condición de parada. Por su parte, *min_delta*=0 establece que cualquier mejora, por pequeña que sea, será considerada válida.

Además, se inicializan varias variables internas necesarias para el control del proceso:

- *self.best_loss*, que almacena el mejor valor de pérdida observado hasta el momento.
- *self.counter*, que contabiliza el número de épocas consecutivas sin mejora.
- *self.early_stop*, una variable booleana que indica si debe detenerse el entrenamiento.

El método *call* recibe en cada época el valor actual de la pérdida de validación (*val_loss*). En la primera evaluación, dicho valor se guarda como referencia inicial. En las siguientes épocas, se compara con el mejor valor registrado. Si la pérdida de validación no disminuye respecto al mejor valor registrado, el contador aumenta. Cuando el número de épocas sin mejora alcanza el valor de *patience*, la variable *early_stop* pasa a ser True, señalando que debe interrumpirse el entrenamiento. En cambio, si la pérdida mejora, se actualiza *best_loss* y el contador se reinicia.

De este modo, la clase implementa un mecanismo autónomo que permite controlar la evolución del entrenamiento en función del comportamiento de la pérdida de validación.

3.5.3 Desarrollo del proceso de entrenamiento

En este apartado se describe la implementación completa del proceso de entrenamiento del modelo. Se integran distintas estrategias orientadas a mejorar la capacidad de generalización, como el uso de validación independiente en cada época, el ajuste dinámico del *learning rate* y la aplicación del mecanismo de parada definido previamente. Asimismo, se incorpora un sistema de guardado del mejor modelo en función de la pérdida de validación.

Antes de iniciar el entrenamiento, se define una configuración básica del proceso. El modelo se entrena durante un máximo de 60 épocas, permitiendo que el proceso de optimización alcance la convergencia. Además, se activa un mecanismo de *Early Stopping* con *patience* = 30 y *min_delta* = 0.001, lo que implica que el entrenamiento se detendrá si la pérdida de validación no mejora al menos 0.001 durante 30 épocas consecutivas.

En cada época, el modelo recibe lotes de imágenes procedentes del conjunto de entrenamiento, formado por las clases dron y ave. Para cada lote se realiza la propagación hacia delante (forward pass), se calcula la pérdida mediante la función definida previamente y se actualizan los pesos del modelo mediante retropropagación (*backpropagation*). La precisión en entrenamiento se calcula comparando la clase predicha por el modelo con la etiqueta real de cada imagen.

Finalizada la fase de entrenamiento, el modelo se evalúa sobre el conjunto de validación utilizando imágenes originales. Esto permite medir el rendimiento real del modelo. La pérdida de validación se utiliza tanto para ajustar dinámicamente el *learning rate* como para determinar si se ha obtenido un nuevo mejor modelo, guardando en ese caso sus pesos.

Al concluir el proceso, ya sea por alcanzar las 60 épocas o por activación del *Early Stopping*, se cargan los pesos correspondientes al mejor rendimiento observado en validación, garantizando que el modelo final no corresponde necesariamente a la última época, sino a la más óptima.

3.5.4 Almacenamiento y carga del modelo entrenado

Una vez finalizado el entrenamiento, se guardan los pesos del modelo en un archivo *.pth*. Posteriormente, se cargan dichos pesos en la arquitectura definida, asegurando que el modelo queda preparado en modo evaluación y en el dispositivo correspondiente (CPU o GPU). De este modo, el modelo entrenado puede reutilizarse sin necesidad de repetir el proceso de entrenamiento.

4 RESULTADOS / VALIDACIÓN / PRUEBA

4.1 Metodología de evaluación del modelo

4.1.1 Evaluación del proceso de entrenamiento (loss y accuracy)

Con el objetivo de analizar el comportamiento del modelo durante el proceso de aprendizaje, se representa la evolución de dos métricas a lo largo de las distintas épocas de entrenamiento: la función de pérdida (*loss*) y la precisión de clasificación (*accuracy*). Estas métricas permiten evaluar cómo mejora el modelo a medida que ajusta sus parámetros y facilitan la detección de posibles problemas durante el entrenamiento.

La función de pérdida mide el error cometido por el modelo al realizar sus predicciones. Durante el entrenamiento, el comportamiento esperado es que esta métrica disminuya progresivamente a medida que la red neuronal aprende a representar mejor las características relevantes de los datos. Para evaluar la capacidad de generalización del modelo, la pérdida se calcula tanto sobre el conjunto de entrenamiento como sobre el conjunto de validación.

Por otro lado, la precisión (*accuracy*) indica el porcentaje de predicciones correctas realizadas por el modelo. A medida que avanza el entrenamiento, se espera que esta métrica aumente progresivamente si el modelo está aprendiendo patrones útiles para la tarea de clasificación.

La comparación entre las curvas de entrenamiento y validación permite además identificar posibles fenómenos de sobreajuste (*overfitting*). Cuando ambas curvas presentan un comportamiento similar y evolucionan de forma paralela, se puede considerar que el modelo generaliza adecuadamente a datos no vistos durante el entrenamiento. En cambio, una divergencia significativa entre ambas curvas podría indicar que el modelo está memorizando el conjunto de entrenamiento en lugar de aprender representaciones generalizables.

Este tipo de representaciones gráficas proporciona por tanto una visión global del proceso de aprendizaje del modelo, permitiendo evaluar su estabilidad, velocidad de convergencia y capacidad de generalización antes de analizar los resultados obtenidos en las etapas de evaluación posteriores.

4.1.2 Análisis cualitativo mediante mapas de activación (Grad-CAM)

Con el objetivo de interpretar el comportamiento interno del modelo y analizar qué regiones de la imagen influyen en la decisión final, se emplea la técnica Grad-CAM (*Gradient-weighted Class Activation Mapping*). Este método permite generar mapas de activación que resaltan las zonas de la imagen que han contribuido de forma más significativa a la predicción realizada por la red neuronal.

El procedimiento consiste en introducir la imagen en la red neuronal para obtener la predicción del modelo y calcular los gradientes asociados a la clase objetivo. A partir de estos gradientes se generan pesos que se combinan con las activaciones de una capa convolucional profunda, produciendo un mapa de calor que destaca las regiones más relevantes para la clasificación.

Posteriormente, el mapa generado se redimensiona al tamaño original de la imagen y se superpone sobre la imagen de entrada, obteniendo una representación visual que permite identificar qué partes de la escena han sido utilizadas por el modelo para tomar su decisión. Este tipo de análisis cualitativo resulta especialmente útil para verificar si el modelo centra su atención en el objeto de interés o si, por el contrario, se ve influenciado por elementos del fondo.

4.1.3 Evaluación cuantitativa mediante matriz de confusión

Con el fin de evaluar de forma cuantitativa el rendimiento del modelo de clasificación, se emplea la matriz de confusión junto con diversas métricas derivadas de ella. Estas herramientas permiten analizar el comportamiento del sistema no solo en términos de precisión global, sino también en relación con cada una de las clases consideradas.

La matriz de confusión es una representación tabular que muestra la relación entre las etiquetas reales de las imágenes y las clases predichas por el modelo. En el caso del problema abordado en este trabajo, que consiste en distinguir entre aves y drones, la matriz permite identificar cuatro tipos de resultados: verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. De este modo, es posible observar de manera directa cuántas imágenes de cada clase han sido clasificadas correctamente y cuántas han sido confundidas con la otra categoría.

A partir de esta matriz se calculan diversas métricas de evaluación que permiten caracterizar con mayor precisión el rendimiento del modelo. Entre ellas se encuentran la precisión **(***precision***)**, que indica la proporción de predicciones correctas entre todas las predicciones realizadas para una determinada clase; el *recall*, que mide la capacidad del modelo para detectar correctamente los elementos pertenecientes a dicha clase; y el F1-score, que combina ambas métricas en una única medida equilibrada.

Asimismo, se calcula la precisión global **(***accuracy***)**, que representa el porcentaje total de predicciones correctas realizadas por el modelo sobre el conjunto de imágenes evaluado. El análisis conjunto de estas métricas proporciona una visión más completa del comportamiento del sistema de clasificación, permitiendo evaluar su capacidad para diferenciar entre drones y aves en distintos escenarios.

Estas métricas complementan el análisis realizado mediante las curvas de entrenamiento y los mapas de activación Grad-CAM, proporcionando una evaluación cuantitativa del rendimiento del modelo en la tarea de clasificación.

Una vez descrita la metodología empleada para evaluar el rendimiento del modelo, en los siguientes apartados se presentan los resultados obtenidos en los distintos experimentos realizados.

4.2 Modelos evaluados sobre el *dataset* de prueba

En este apartado se presentan los resultados obtenidos al evaluar los distintos modelos utilizando el conjunto de datos de prueba procedente de los *datasets* empleados durante el desarrollo del trabajo.

4.2.1 Modelo sin Sobel

En este apartado se analizan los resultados obtenidos al aplicar el modelo sin la utilización del filtro Sobel sobre el conjunto de prueba del *dataset*. El análisis se realiza siguiendo los tres criterios descritos previamente: la evolución del entrenamiento mediante las métricas de pérdida (*loss*) y precisión (*accuracy*), el análisis cualitativo mediante mapas de activación Grad-CAM y la evaluación cuantitativa a través de la matriz de confusión y las métricas de clasificación correspondientes.

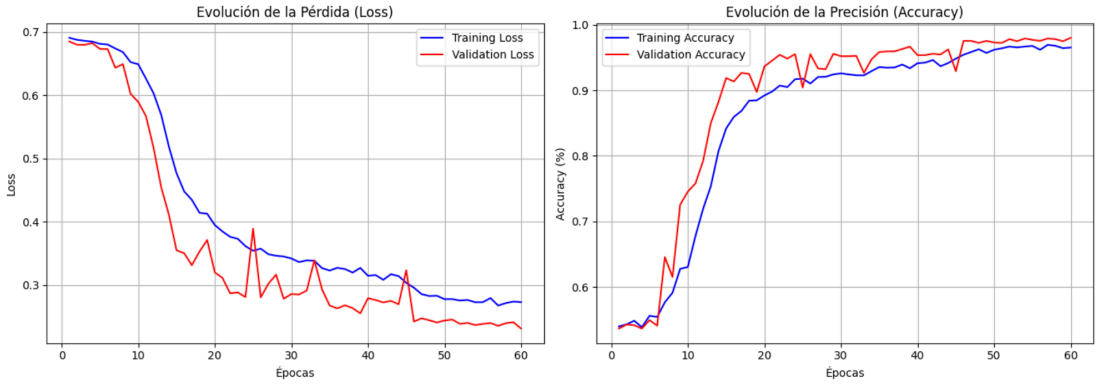


Figura 4-1 Evolución de la pérdida (izquierda) y precisión (derecha) del modelo sin Sobel durante el entrenamiento (fuente: propia)

En la Figura 4-1 se muestra la evolución de la pérdida y la precisión del modelo durante el entrenamiento. La pérdida disminuye progresivamente tanto en el conjunto de entrenamiento como en el de validación, mientras que la precisión aumenta de forma gradual hasta estabilizarse en valores cercanos al 96–97 %.

La evolución conjunta de las curvas de entrenamiento y validación sugiere un proceso de aprendizaje estable, sin evidencias claras de sobreajuste (*overfitting*). La proximidad entre ambas curvas indica que el modelo es capaz de generalizar adecuadamente a datos no vistos durante el entrenamiento, lo que refleja un proceso de entrenamiento efectivo y un conjunto de datos suficientemente representativo para la tarea de clasificación entre aves y drones.

A continuación, se presentan algunos ejemplos representativos de los mapas de activación Grad-CAM obtenidos al aplicar el modelo sin filtro Sobel sobre imágenes del conjunto de prueba. Estas visualizaciones permiten analizar de forma cualitativa en qué regiones de la imagen centra su atención el modelo al realizar la clasificación entre aves y drones.

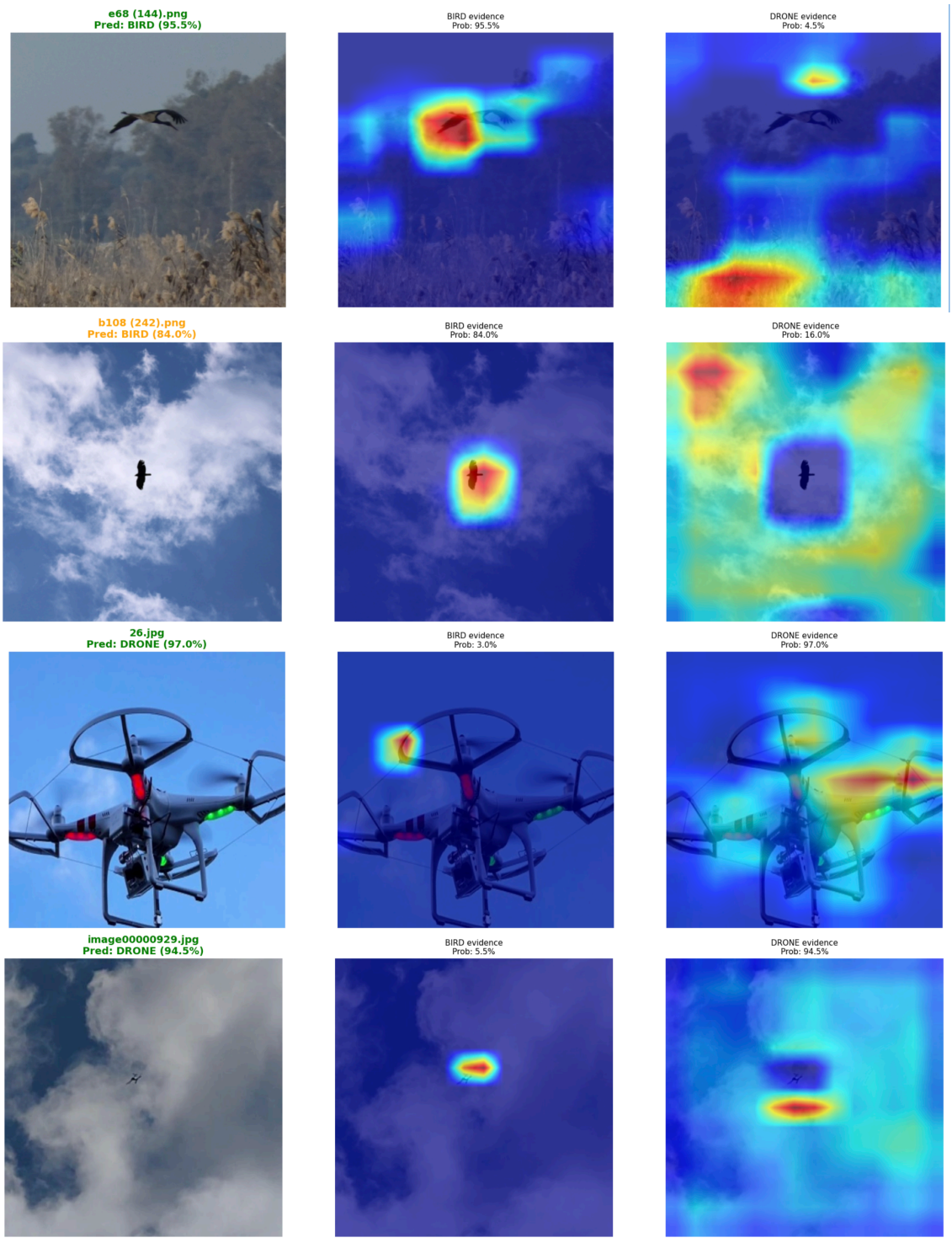


Figura 4-2 Ejemplos de mapas Grad-CAM generados por el modelo sin Sobel. De izquierda a derecha: imagen original, mapa de activación para la clase ave y mapa de activación para la clase dron (fuente: propia)

En los primeros ejemplos de la Figura 4-2 , correspondientes a imágenes de aves, puede observarse cómo el mapa Grad-CAM concentra su activación principalmente sobre el cuerpo y las alas del animal. Esto indica que el modelo es capaz de identificar características estructurales propias de las aves y utilizarlas para realizar la clasificación. No obstante, en algunos casos también aparecen activaciones en zonas del fondo de la escena, lo que sugiere que el modelo puede estar utilizando parcialmente información contextual del entorno.

En el segundo ejemplo, donde el ave aparece sobre un fondo de cielo con nubes, la activación principal se sitúa sobre la silueta del animal, aunque pueden observarse activaciones adicionales en regiones del fondo con menor intensidad. Este comportamiento sugiere que, aunque el modelo ha aprendido a reconocer la forma característica del ave, ciertos elementos del entorno también pueden influir en la decisión final.

Los ejemplos correspondientes a drones muestran un patrón similar. En la tercera imagen, donde el dron aparece a corta distancia, las zonas de mayor activación se distribuyen sobre la estructura del dispositivo, incluyendo el cuerpo central y parte de los brazos. Sin embargo, también pueden observarse activaciones secundarias en otras zonas de la imagen que no corresponden directamente al objeto.

Por último, en el caso del dron situado a mayor distancia dentro de la escena, el mapa de activación continúa señalando la posición del objeto, aunque aparecen regiones adicionales de activación en el fondo. Esto indica que el modelo es capaz de localizar el dron incluso cuando ocupa una región pequeña de la imagen, pero también pone de manifiesto que parte de la decisión puede apoyarse en características contextuales presentes en la escena.

En conjunto, estos ejemplos muestran que el modelo identifica correctamente las regiones donde se encuentran los objetos principales, aunque en determinados casos también se observan activaciones en elementos del fondo. Este comportamiento sugiere que el modelo puede estar utilizando tanto características del objeto como información contextual de la escena para realizar la clasificación.

A continuación, se presentan los resultados obtenidos al evaluar el modelo sin la aplicación del filtro Sobel sobre el conjunto de prueba del *dataset*. Para ello, se analiza la matriz de confusión obtenida (Figura 4-3) junto con las métricas de clasificación correspondientes (Tabla 4-1).

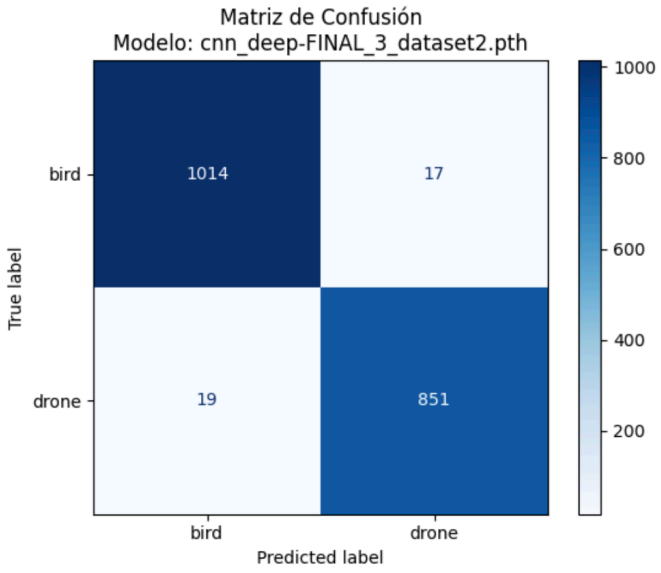


Figura 4-3 Matriz de confusión del modelo sin filtro Sobel (fuente: propia)

El modelo clasifica correctamente la gran mayoría de las imágenes de ambas clases. En el caso de las aves, 1014 de las 1031 imágenes son identificadas correctamente, mientras que únicamente 17 son clasificadas erróneamente como drones. De forma similar, para la clase dron se obtienen 851 clasificaciones correctas de un total de 870 imágenes, habiendo únicamente 19 errores.

Clase	Precisión	*Recall*	*F1-score*	Soporte
Pájaro	0.98	0.98	0.98	1031
Dron	0.98	0.98	0.98	870
Accuracy global			0.9811	1901

Tabla 4-1 Métricas de clasificación del modelo sin Sobel

Estos resultados se reflejan también en las métricas de clasificación, donde tanto la precisión (*precision*), el *recall* y el *F1-score* alcanzan valores de 0.98 para ambas clases. Asimismo, la precisión global del modelo alcanza un valor del 98.11 %, lo que indica un rendimiento muy elevado en la tarea de clasificación entre aves y drones.

En conjunto, la matriz de confusión y las métricas obtenidas muestran que el modelo presenta un comportamiento equilibrado entre ambas clases, con un número muy reducido de errores de clasificación y una elevada capacidad para diferenciar entre ambos tipos de objetos.

4.2.2 Modelo con Sobel

En este apartado se presentan los resultados obtenidos al aplicar el modelo que incorpora el filtro Sobel sobre el conjunto de prueba del *dataset*. El análisis se realiza siguiendo los mismos criterios descritos anteriormente, permitiendo comparar su comportamiento con el modelo sin Sobel.

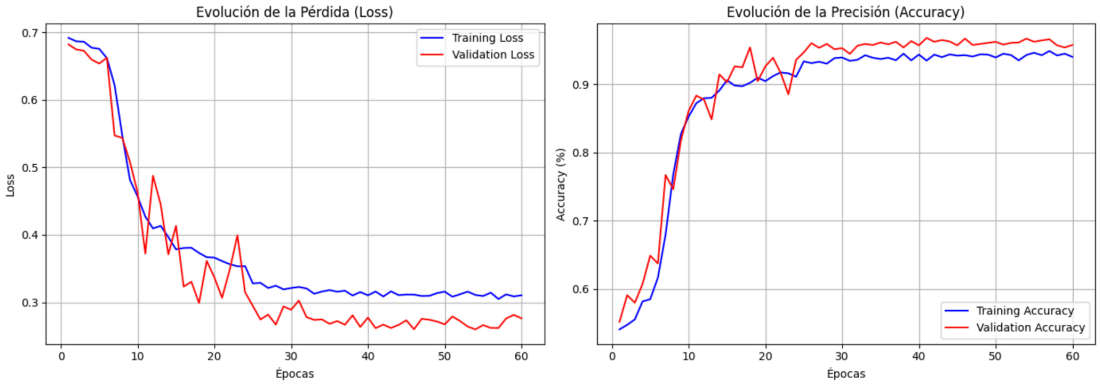


Figura 4-4 Evolución de la pérdida (izquierda) y precisión (derecha) del modelo con Sobel durante el entrenamiento (fuente: propia)

En la Figura 4-4 se presenta la evolución de la pérdida y la precisión del modelo con Sobel a lo largo del entrenamiento. La pérdida disminuye progresivamente mientras que la precisión aumenta durante las primeras épocas hasta estabilizarse en torno al 95 %. El comportamiento similar entre las curvas de entrenamiento y validación indica un proceso de aprendizaje estable y sin indicios claros de sobreajuste.

A continuación, se presentan algunos ejemplos representativos de los mapas de activación Grad-CAM obtenidos con el modelo que incorpora el filtro Sobel (Figura 4-5).

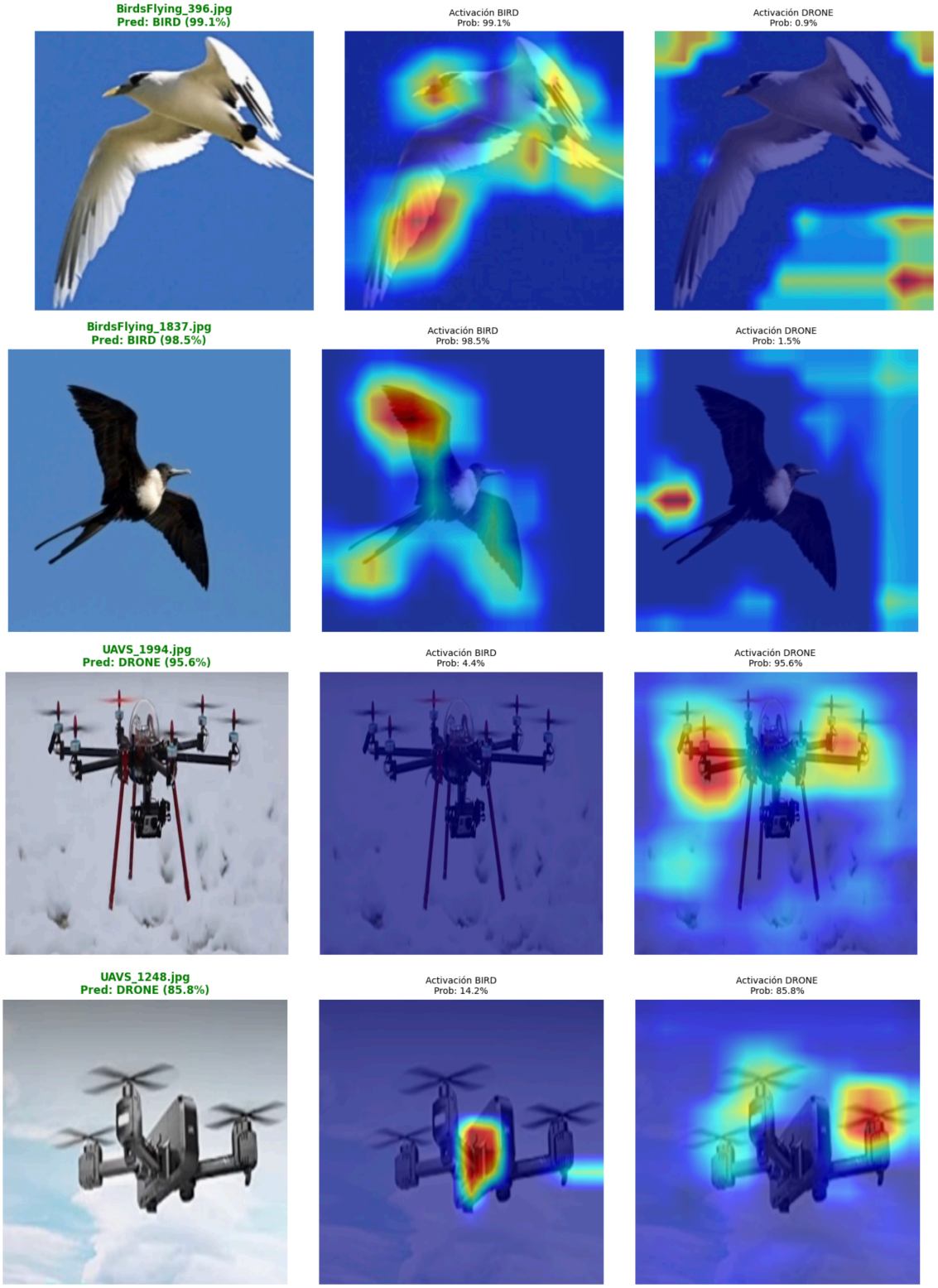


Figura 4-5 Ejemplos de mapas Grad-CAM generados por el modelo con Sobel. De izquierda a derecha: imagen original, mapa de activación para la clase ave y mapa de activación para la clase dron (fuente: propia)

En los ejemplos correspondientes a aves, el mapa de activación se concentra principalmente sobre el cuerpo y las alas del animal. Las regiones de mayor intensidad coinciden con las estructuras

principales del ave, mientras que el fondo de la escena presenta activaciones mucho más débiles. Esto indica que el modelo centra su atención en las características estructurales relevantes del objeto.

En el caso de los drones, los mapas Grad-CAM muestran activaciones distribuidas sobre el cuerpo central del dispositivo y sobre los brazos donde se encuentran las hélices. Estas zonas corresponden a elementos geométricos característicos de este tipo de aeronaves, lo que sugiere que el modelo utiliza dichas estructuras para realizar la clasificación.

En conjunto, los ejemplos analizados indican que el modelo es capaz de localizar las regiones relevantes de los objetos presentes en la imagen y basar su decisión en características visuales discriminativas.

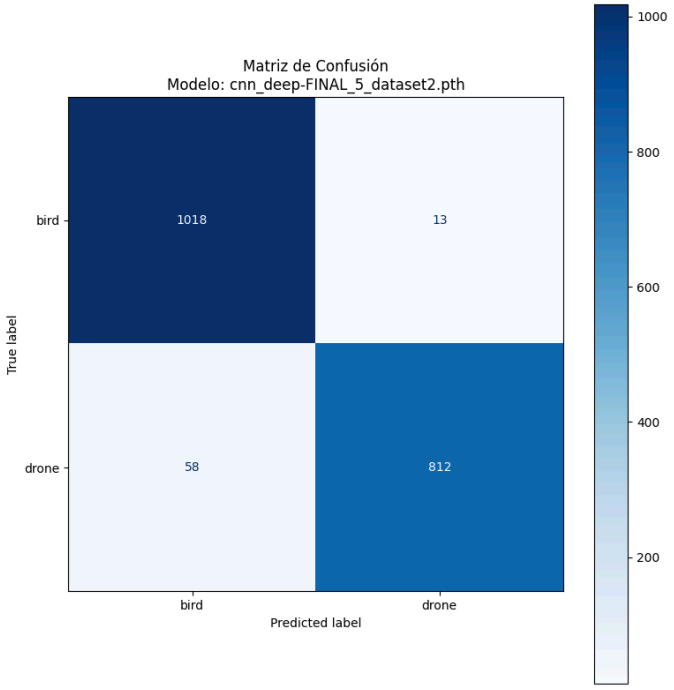


Figura 4-6 Matriz de confusión del modelo con filtro Sobel (fuente: propia)

La Figura 4-6 muestra la matriz de confusión obtenida al evaluar el modelo con Sobel sobre el conjunto de prueba. En el caso de la clase ave, 1018 de las 1031 imágenes son clasificadas correctamente, mientras que únicamente 13 se confunden con la clase dron.

Para la clase dron se obtienen 812 clasificaciones correctas de un total de 870 imágenes, produciéndose 58 errores en los que el modelo identifica el dron como ave. Aunque el número de errores es ligeramente superior en esta clase, la mayoría de las imágenes siguen siendo clasificadas correctamente.

Clase	Precisión	*Recall*	*F1-score*	Soporte
Pájaro	0.95	0.99	0.97	1031
Dron	0.98	0.93	0.96	870
Accuracy global			0.96	1901

Tabla 4-2 Métricas de clasificación del modelo sin Sobel

Las métricas de clasificación obtenidas (Tabla 4-2) reflejan este comportamiento. La clase ave alcanza valores de precisión de 0.95, *recall* de 0.99 y un F1-score de 0.97. Por su parte, la clase dron presenta una precisión de 0.98, un *recall* de 0.93 y un F1-score de 0.96.

La precisión global del modelo alcanza un valor de 96.27 %, lo que indica un rendimiento elevado en la tarea de clasificación entre aves y drones, aunque ligeramente inferior al obtenido por el modelo sin Sobel.

4.2.3 Modelo Transfer-learning

En este apartado se presentan los resultados obtenidos al aplicar el modelo *transfer-learning* sobre el conjunto de prueba del *dataset*. El análisis se realiza siguiendo los mismos criterios descritos anteriormente, permitiendo comparar su comportamiento con los modelos con y sin Sobel.

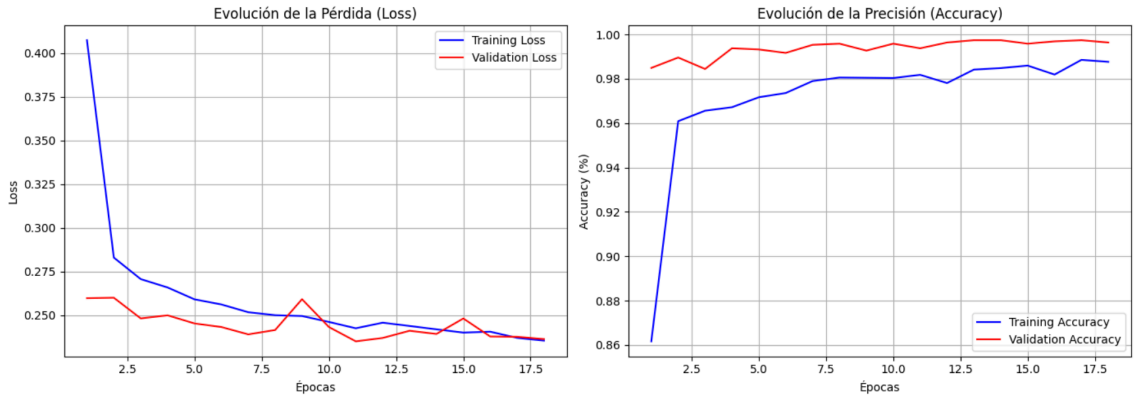


Figura 4-7 Evolución de la pérdida (izquierda) y precisión (derecha) del modelo *transfer-learning* durante el entrenamiento (fuente: propia)

La evolución de las curvas vista en la Figura 4-7 ,sigue un patrón similar al observado en los modelos anteriores. La pérdida disminuye progresivamente mientras que la precisión aumenta durante las primeras épocas hasta estabilizarse. La proximidad entre las curvas de entrenamiento y validación indica un proceso de aprendizaje estable y sin evidencias claras de sobreajuste (*overfitting*).

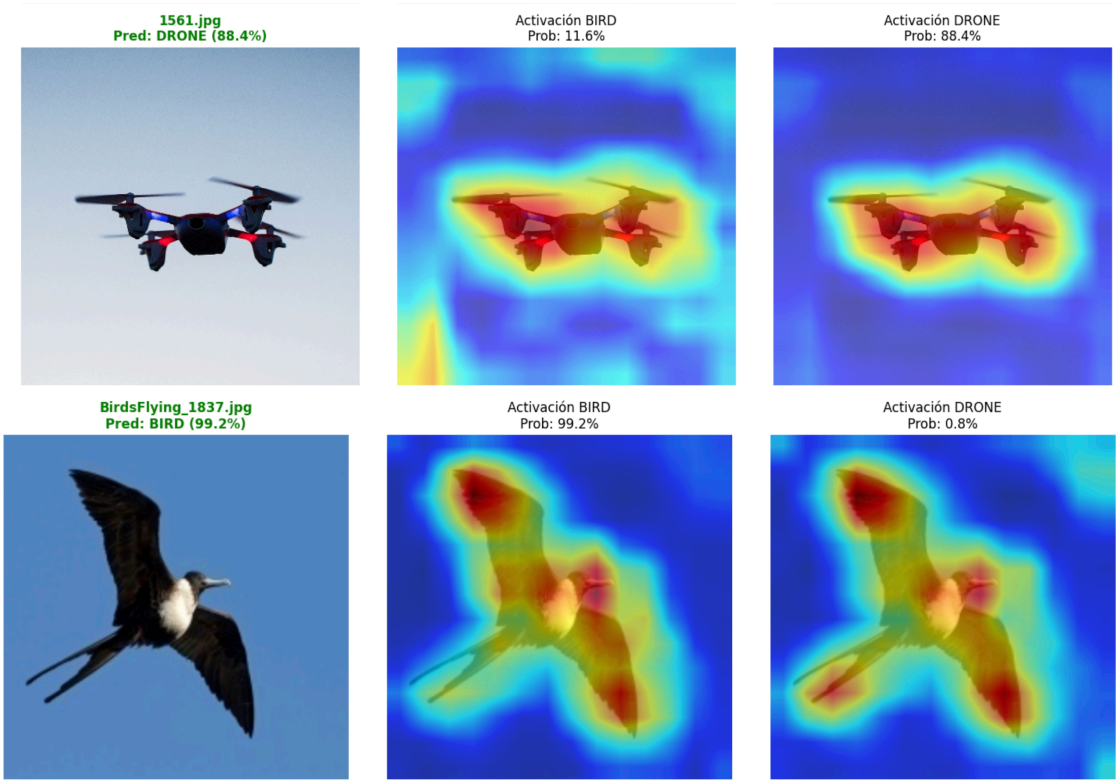


Figura 4-8 Ejemplos de mapas Grad-CAM generados por el modelo transfer-learning. De izquierda a derecha: imagen original, mapa de activación para la clase ave y mapa de activación para la clase dron (fuente: propia)

En ambos casos de la Figura 4-8 las activaciones se concentran principalmente sobre la región del objeto detectado, abarcando la totalidad de cada silueta. En el caso del dron, la activación se distribuye a lo largo de la estructura y los brazos del dispositivo, mientras que en la imagen del ave se observa una activación amplia que cubre el cuerpo y las alas, observándose poca activación en los fondos de las imágenes. Este comportamiento sugiere que el modelo utiliza principalmente características globales del objeto para realizar la predicción.

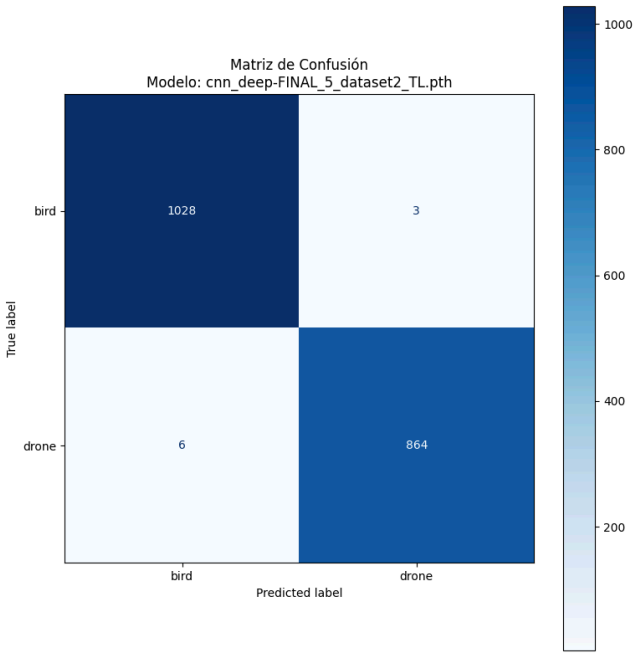


Figura 4-9 Matriz de confusión del modelo *transfer-learning* (fuente: propia)

La matriz de confusión (Figura 4-9) refleja un elevado nivel de acierto en la clasificación de ambas clases. La gran mayoría de las imágenes de aves y drones son clasificadas correctamente, registrándose únicamente un número muy reducido de errores. Esto indica que el modelo basado en *transfer learning* es capaz de distinguir eficazmente entre ambas categorías dentro del conjunto de datos de prueba.

Clase	**Precisión**	***Recall***	***F1-score***	**Soporte**
Pájaro	0.99	1.00	1.00	1031
Dron	1.00	0.99	0.99	870
Accuracy global			1.00	1901

Tabla 4-3 Métricas de clasificación del modelo *transfer-learning*

Tal y como se observa en la Tabla 4-3 ,tanto la precisión como el *recall* y el F1-score alcanzan valores muy elevados para ambas clases, lo que confirma el buen rendimiento del modelo en la tarea de clasificación. En conjunto, el sistema alcanza una precisión global (*accuracy*) del 99,53 % sobre el conjunto de datos evaluado. Estos resultados sugieren que el modelo basado en *transfer-learning* logra aprender de forma eficaz las características necesarias para diferenciar entre aves y drones en el conjunto de datos utilizado, mostrando un comportamiento muy fiable durante la evaluación.

4.3 Prueba con imágenes propias

En este apartado se presentan los resultados obtenidos al evaluar los diferentes modelos utilizando un conjunto de imágenes propias, con el objetivo de analizar su comportamiento en un escenario más cercano a una situación real. Estas imágenes han sido tomadas y seleccionadas manualmente y representan diferentes situaciones reales en las que aparecen aves o drones en distintos contextos y condiciones visuales.

El objetivo de esta evaluación es analizar la capacidad de generalización del modelo cuando se enfrenta a imágenes no presentes en el conjunto de entrenamiento, permitiendo observar tanto las predicciones obtenidas como las regiones de la imagen que influyen en la decisión del modelo mediante mapas de activación Grad-CAM.

4.3.1 Modelo sin Sobel

En este apartado se evalúa el comportamiento del modelo sin la aplicación del filtro Sobel utilizando un conjunto de imágenes propias, externas al *dataset* empleado durante el entrenamiento.

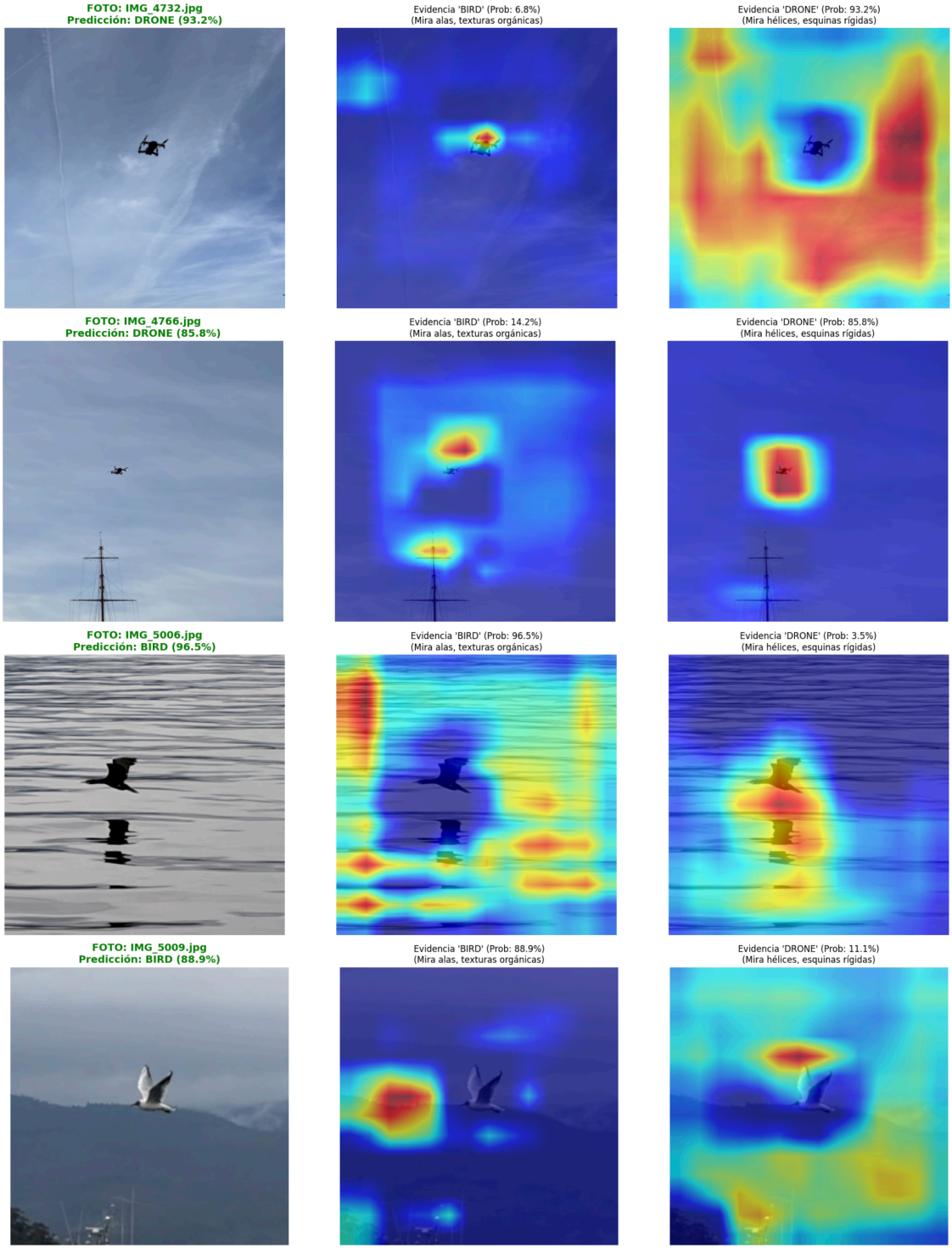


Figura 4-10 Ejemplos de mapas Grad-CAM generados por el modelo sin Sobel a imágenes propias. De izquierda a derecha: imagen original, mapa de activación para la clase ave y mapa de activación para la clase dron (fuente: propia)

En los ejemplos de la Figura 4-10 se puede observar cómo el modelo es capaz de localizar el objeto principal en la mayoría de los casos, generando activaciones concentradas alrededor del dron o del ave presente en la escena. En las dos primeras imágenes, correspondientes a drones, el mapa Grad-CAM muestra una activación clara sobre el objeto detectado, indicando que el modelo identifica correctamente la región donde se encuentra el dron dentro de la imagen.

Sin embargo, también se observan activaciones en otras zonas del fondo, especialmente en regiones amplias del cielo o en estructuras presentes en la escena. Este comportamiento puede apreciarse en el segundo ejemplo, donde además de la activación sobre el dron aparece una zona destacada en la parte inferior de la imagen correspondiente al mástil del barco.

En las imágenes correspondientes a aves ocurre algo similar. Aunque el modelo genera activaciones sobre el cuerpo y las alas del ave, también aparecen zonas de activación en regiones del fondo, como la superficie del agua o partes del paisaje. Este fenómeno sugiere que el modelo puede estar utilizando información contextual del entorno además de las características propias del objeto para realizar la clasificación.

En conjunto, estos resultados indican que, aunque el modelo es capaz de identificar correctamente el objeto en la imagen, en algunos casos también presta atención a elementos del fondo que podrían estar correlacionados con alguna de las clases dentro del *dataset*. Este comportamiento puede contribuir a obtener métricas elevadas durante la evaluación sobre el conjunto de prueba, pero también puede limitar la capacidad de generalización del modelo cuando se enfrenta a imágenes de entornos diferentes.

Para evaluar cuantitativamente el rendimiento del modelo en las pruebas realizadas con imágenes externas al *dataset*, se analiza la matriz de confusión obtenida a partir de las 68 imágenes evaluadas (Figura 4-11).

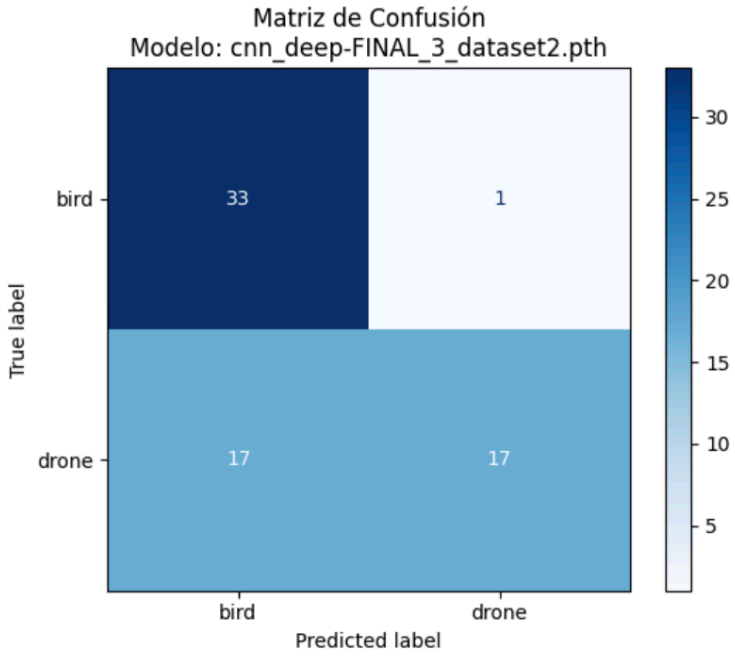


Figura 4-11 Matriz de confusión del modelo sin filtro Sobel para prueba con imágenes propias (fuente: propia)

A partir de la matriz se observa que el modelo identifica correctamente la mayoría de las imágenes de aves, clasificando correctamente 33 de las 34 imágenes pertenecientes a esta clase y confundiendo únicamente una de ellas con la categoría de dron. Este comportamiento se refleja en un valor elevado de *recall* para la clase *bird*, que alcanza 0.97.

Sin embargo, el comportamiento del modelo es menos preciso en el caso de los drones. De las 34 imágenes correspondientes a esta clase, únicamente 17 son clasificadas correctamente como drones, mientras que las otras 17 son clasificadas erróneamente como aves. Esta situación provoca un descenso significativo en el *recall* de la clase *drone*, que se sitúa en 0.50. Las métricas de clasificación del modelo sin Sobel para prueba con imágenes propias se ven en la Tabla 4-4 .

Clase	Precisión	*Recall*	*F1-score*	Soporte
Pájaro	0.66	0.97	0.79	34
Dron	0.94	0.50	0.65	34
Accuracy global			0.74	68

Tabla 4-4 Métricas de clasificación del modelo sin Sobel para prueba con imágenes propias

En conjunto, el modelo alcanza una precisión global del 73.53 % en estas pruebas realizadas con imágenes externas al *dataset*. Este resultado es notablemente inferior al obtenido durante la evaluación sobre el conjunto de prueba del *dataset*, lo que sugiere que el modelo presenta dificultades para generalizar correctamente cuando se enfrenta a imágenes capturadas en condiciones diferentes a las presentes durante el entrenamiento.

Este comportamiento refuerza la importancia de evaluar los modelos de visión por computador no solo mediante métricas obtenidas sobre el *dataset* de prueba, sino también mediante pruebas con imágenes reales o externas al conjunto de entrenamiento, ya que estas permiten analizar con mayor precisión la capacidad de generalización del sistema.

4.3.2 Modelo con Sobel

En este apartado se evalúa el comportamiento del modelo con la aplicación del filtro Sobel utilizando un conjunto de imágenes propias, externas al *dataset* empleado durante el entrenamiento.

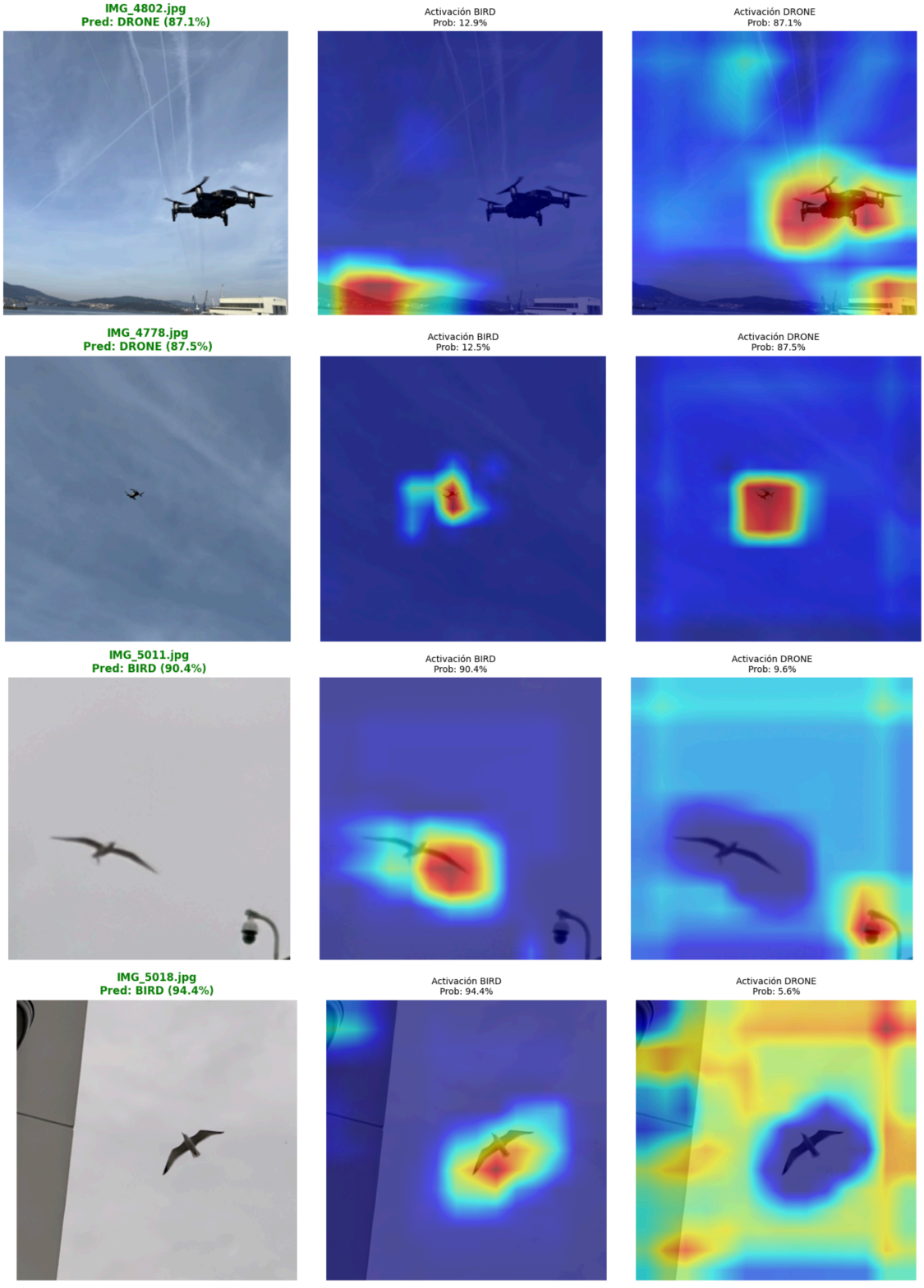


Figura 4-12 Ejemplos de mapas Grad-CAM generados por el modelo con Sobel a imágenes propias. De izquierda a derecha: imagen original, mapa de activación para la clase ave y mapa de activación para la clase dron (fuente: propia)

En las dos primeras imágenes de la Figura 4-12 , correspondientes a drones, se observa que las activaciones del mapa Grad-CAM se concentran principalmente sobre el propio objeto. En particular, en el segundo ejemplo el modelo identifica con claridad la región donde se encuentra el dron, generando una activación compacta centrada en la estructura del aparato. Este comportamiento indica que la red

neuronal está utilizando características propias del objeto, como su forma y contornos, para realizar la clasificación.

En el caso de las imágenes correspondientes a aves de la Figura 4-12 , el modelo también muestra activaciones claramente localizadas sobre el cuerpo y las alas del animal. En ambos ejemplos, el mapa de activación se concentra principalmente sobre la silueta del ave, mientras que las regiones del fondo presentan una influencia significativamente menor en la decisión final del modelo.

Aunque en algunos casos pueden aparecer pequeñas activaciones en otras zonas de la imagen, estas son considerablemente menos intensas que las observadas sobre el objeto principal. En conjunto, los ejemplos analizados muestran que el modelo que incorpora el filtro Sobel tiende a centrar su atención en las estructuras del objeto de interés, reduciendo la dependencia de elementos del fondo presentes en la escena.

Este comportamiento resulta especialmente relevante en el contexto de las pruebas realizadas con imágenes externas al *dataset*, ya que sugiere que la incorporación del filtro Sobel favorece que la red neuronal aprenda representaciones más relacionadas con la forma y los contornos del objeto, lo que contribuye a mejorar la capacidad de generalización del modelo en escenarios reales.

Para completar la evaluación cuantitativa del modelo en las pruebas realizadas con imágenes propias, se analiza la matriz de confusión obtenida al aplicar el modelo que incorpora el filtro Sobel sobre el mismo conjunto de 68 imágenes (Figura 4-13).

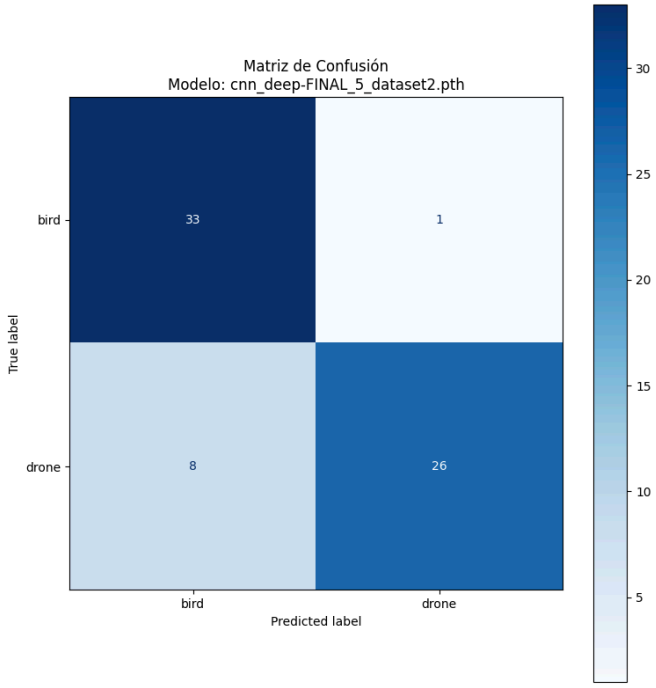


Figura 4-13 Matriz de confusión del modelo con filtro Sobel para prueba con imágenes propias (fuente: propia)

A partir de la matriz de confusión se observa que el modelo identifica correctamente 33 de las 34 imágenes correspondientes a la clase *bird*, confundiendo únicamente una de ellas con la clase *drone*. Este comportamiento se refleja en un valor elevado de *recall* para esta clase, que alcanza 0.97, lo que indica que el modelo presenta una alta capacidad para detectar correctamente las aves presentes en las imágenes.

En el caso de los drones, el modelo clasifica correctamente 26 de las 34 imágenes pertenecientes a esta categoría, mientras que 8 de ellas son clasificadas erróneamente como aves. Aunque todavía se producen algunas confusiones entre ambas clases, el número de aciertos es considerablemente mayor que en el modelo sin Sobel.

Clase	Precisión	*Recall*	*F1-score*	Soporte
Pájaro	0.80	0.97	0.88	34
Dron	0.96	0.76	0.85	34
Accuracy global			0.87	68

Tabla 4-5 Métricas de clasificación del modelo con Sobel para prueba con imágenes propias

La Tabla 4-5 muestra como en conjunto, el modelo alcanza una precisión global del 86.76 % en estas pruebas realizadas con imágenes externas al *dataset*, lo que supone una mejora significativa respecto al modelo sin Sobel evaluado previamente sobre el mismo conjunto de imágenes.

Este resultado refuerza la idea de que la incorporación del filtro Sobel contribuye a mejorar el comportamiento del modelo en escenarios reales, favoreciendo que la red neuronal utilice características estructurales del objeto, como sus contornos y formas, en lugar de depender en mayor medida de elementos del fondo presentes en la imagen.

4.3.3 Modelo Transfer-learning

En este apartado se evalúa el comportamiento del modelo *transfer-learning* utilizando un conjunto de imágenes propias, externas al *dataset* empleado durante el entrenamiento.

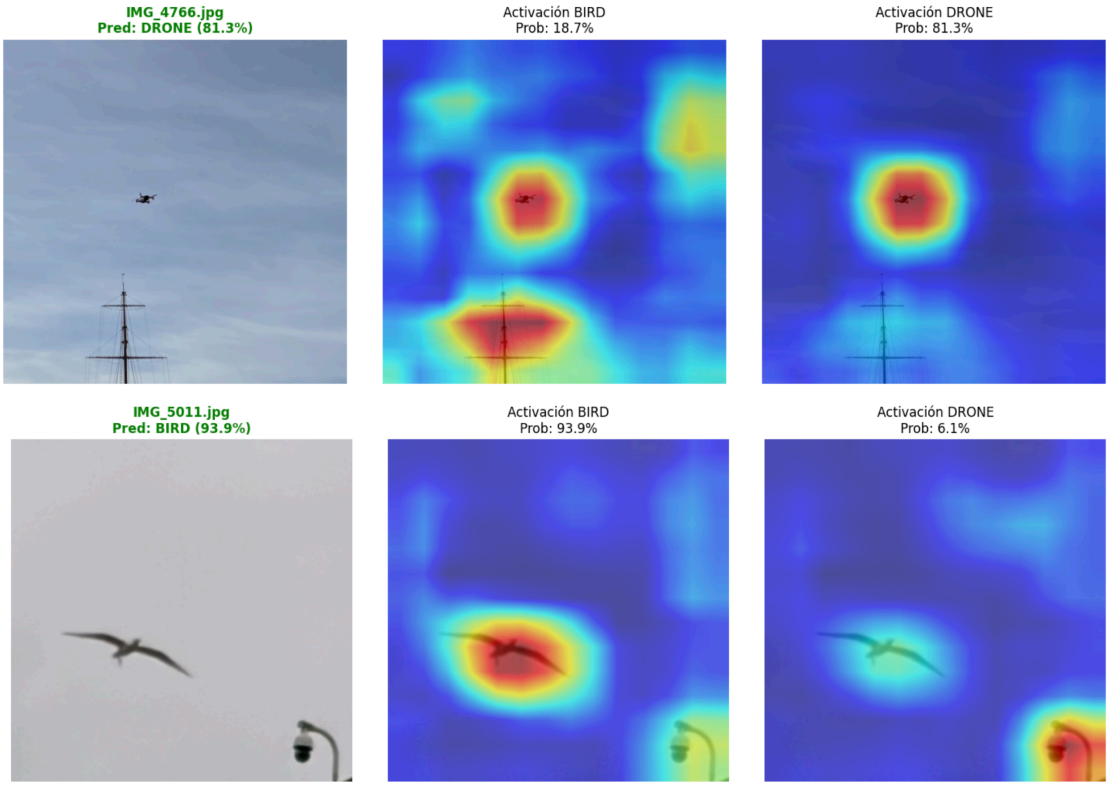


Figura 4-14 Ejemplos de mapas Grad-CAM generados por el modelo *transfer-learning* a imágenes propias. De izquierda a derecha: imagen original, mapa de activación para la clase ave y mapa de activación para la clase dron (fuente: propia)

En la Figura 4-14 se observa que la activación del modelo se concentra principalmente en la región central del objeto detectado. En el caso del dron, la activación se presenta como una región difusa

alrededor del objeto, mientras que en la imagen del pájaro la activación se distribuye sobre gran parte de la silueta.

Estos mapas de activación indican que el modelo identifica correctamente la presencia de objetos en la escena y concentra su atención en la región donde se encuentran, lo que sugiere que está utilizando principalmente características visuales generales del objeto presente en la imagen para realizar la predicción.

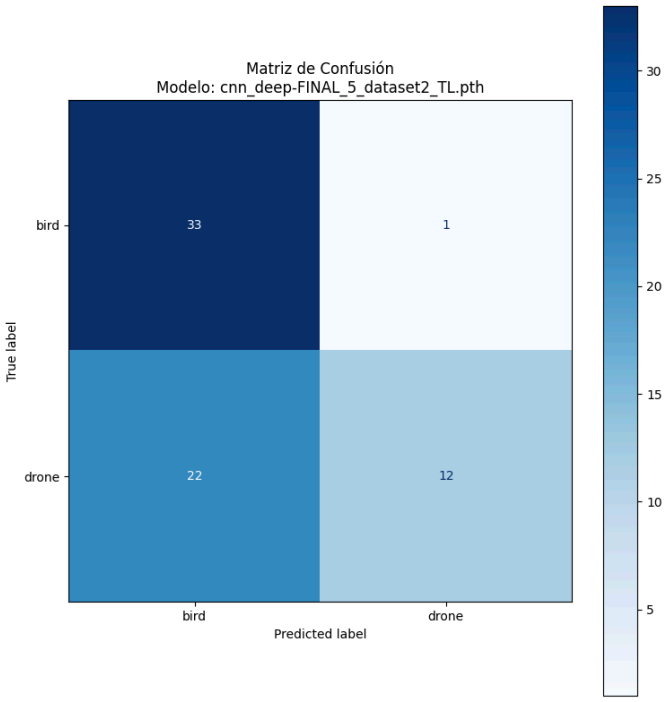


Figura 4-15 Matriz de confusión del modelo *transfer-learning* para prueba con imágenes propias (fuente: propia)

Al evaluar el modelo sobre el conjunto de imágenes propias, se obtuvo una precisión global del 66.18 %. El informe de clasificación muestra un comportamiento desigual entre clases: mientras que la clase *bird* presenta un *recall* elevado (0.97), la clase *drone* obtiene un valor considerablemente menor (0.35). Este resultado, incluido en la Tabla 4-6 , indica que una parte significativa de los drones es clasificada incorrectamente como pájaros.

Clase	**Precisión**	***Recall***	***F1-score***	**Soporte**
Pájaro	0.60	0.97	0.74	34
Dron	0.92	0.35	0.51	34
Accuracy global			0.66	68

Tabla 4-6 Métricas de clasificación del modelo *transfer-learning* para prueba con imágenes propias

La matriz de confusión (Figura 4-15) confirma este comportamiento, mostrando que un número elevado de imágenes de drones se asignan a la clase *bird*.

4.4 Discusión de resultados

Los resultados obtenidos en los diferentes experimentos permiten comparar el comportamiento de los modelos entrenados desde cero, tanto sin la incorporación del filtro Sobel como con él, así como el

modelo basado en *transfer-learning*. La Tabla 4-7 resume las métricas de precisión global obtenidas en el conjunto de prueba del *dataset* y en las pruebas realizadas con imágenes propias.

Modelo	***Dataset* test**	**Pruebas reales**
Sin Sobel	98.1%	73.5%
Con Sobel	96.2%	86.7%
Transfer-Learning	99.5%	66.2%

Tabla 4-7 Comparación de la precisión de los diferentes modelos en el *dataset* de prueba y en pruebas reales

Como se observa, el modelo sin Sobel alcanza una precisión superior en el conjunto de prueba del *dataset*, con un valor del 98.1 %, frente al 96.2 % obtenido por el modelo que incorpora el filtro Sobel. Sin embargo, cuando ambos modelos se evalúan utilizando imágenes externas, el comportamiento se invierte: el modelo sin Sobel obtiene una precisión del 73.5 %, mientras que el modelo con Sobel alcanza un 86.7 %, mostrando una mejora significativa en este escenario.

El modelo basado en *transfer-learning*, por su parte, presenta un comportamiento diferente. Aunque durante el entrenamiento alcanza valores muy elevados de precisión (99.58%), al evaluarse sobre el conjunto de imágenes propias obtiene una precisión global del 66.18 %. El informe de clasificación muestra además un *recall* muy alto para la clase pájaro y considerablemente menor para la clase dron, lo que indica que una parte significativa de los drones es clasificada erróneamente como pájaros.

El análisis cualitativo mediante mapas de activación Grad-CAM ayuda a explicar estas diferencias de comportamiento entre los modelos. En el modelo sin Sobel, las activaciones suelen concentrarse sobre el objeto principal, aunque en varios ejemplos se observan también regiones de activación en elementos del fondo de la imagen, como el cielo, el agua o estructuras presentes en la escena. Esto sugiere que el modelo puede estar utilizando información contextual del entorno para apoyar la clasificación, lo que puede resultar beneficioso cuando las imágenes del conjunto de prueba presentan características similares a las del *dataset* original.

No obstante, esta dependencia del contexto ha podido afectar negativamente a la capacidad de generalización del modelo cuando se enfrenta a imágenes capturadas en condiciones diferentes. En las pruebas realizadas con imágenes propias se observa que el modelo sin Sobel presenta mayores dificultades para identificar correctamente algunos drones, clasificándolos erróneamente como aves.

Por el contrario, el modelo que incorpora el filtro Sobel muestra un patrón diferente en los mapas Grad-CAM. En este caso, las activaciones tienden a concentrarse de forma más clara sobre las estructuras principales de los objetos, como las alas de las aves o los brazos y hélices de los drones. Esto indica que la red neuronal está utilizando en mayor medida características estructurales relacionadas con los contornos y la forma de los objetos, reduciendo la influencia de elementos del fondo presentes en la imagen.

Este comportamiento se traduce en una mejora notable en las pruebas realizadas con imágenes propias, donde el modelo con Sobel demuestra una mayor robustez frente a cambios en el entorno o en las condiciones de captura de las imágenes.

Las diferencias observadas pueden apreciarse notablemente al comparar los mapas Grad-CAM del modelo con Sobel y del modelo basado en *transfer-learning*. En el caso del modelo con Sobel, las activaciones se observan en zonas características como los brazos del dron o la forma de las alas o pico en el caso de las aves, como se ha comentado. En cambio, el modelo basado en *transfer-learning* presenta activaciones más concentradas en la región central del objeto, con una distribución más difusa y general

que no refleja claramente la geometría completa del mismo. Se incluyen dos ejemplos claros de esta comparación en la Figura 4-16 .

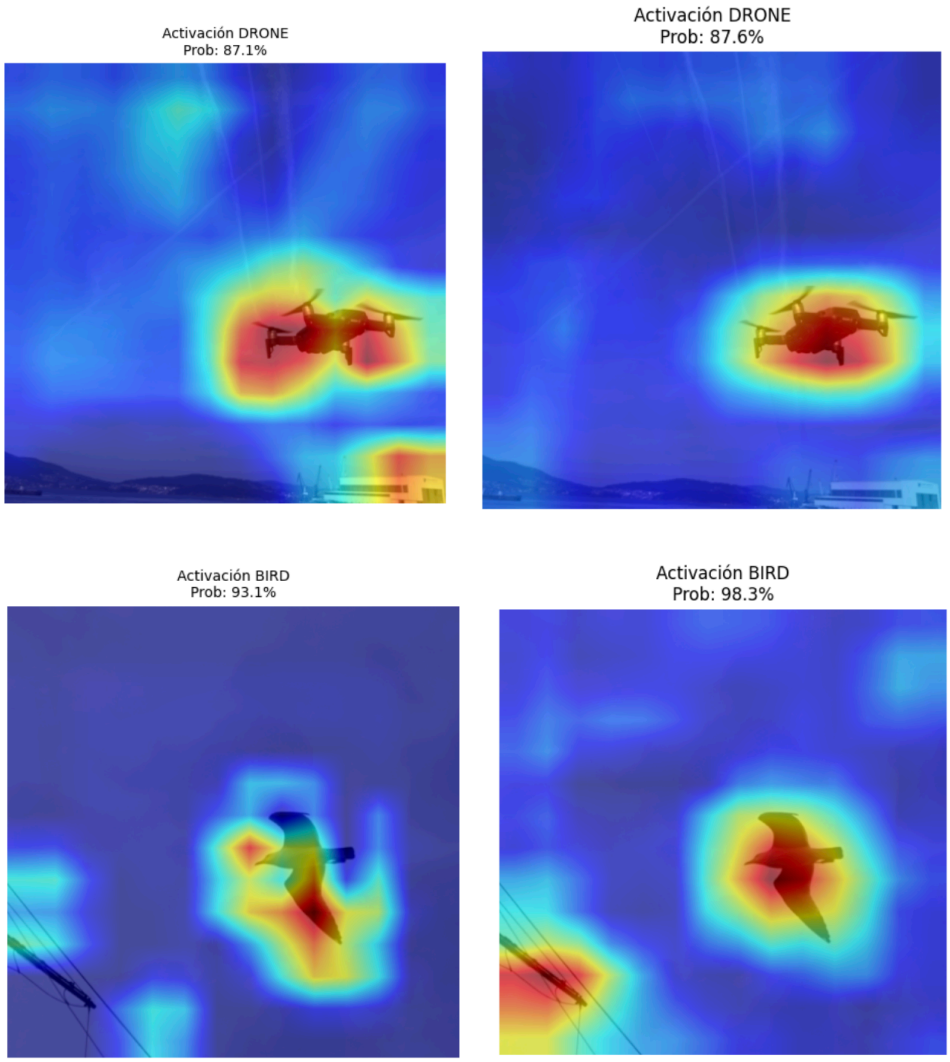


Figura 4-16 Comparación de mapas Grad-CAM entre el modelo con Sobel (izquierda) y el modelo con *transfer-learning* (derecha) (fuente: propia)

Este patrón sugiere que el modelo preentrenado está utilizando principalmente características visuales más generales, como el contraste entre el objeto y el fondo, en lugar de detalles estructurales específicos de cada clase. Esta diferencia puede contribuir a explicar el menor rendimiento observado en las pruebas con imágenes del mundo real.

En conjunto, los resultados obtenidos muestran que la evaluación del rendimiento de un modelo de visión por computador no debe basarse únicamente en las métricas obtenidas sobre el conjunto de prueba del *dataset*. El análisis cualitativo mediante técnicas como Grad-CAM y la evaluación con imágenes externas permiten obtener una visión más completa del comportamiento del modelo y de su capacidad de generalización en escenarios reales.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

El objetivo principal de este trabajo ha sido el desarrollo de un modelo de *Deep Learning* capaz de clasificar imágenes distinguiendo entre drones y aves. Este problema presenta un interés relevante en el ámbito de la vigilancia y defensa aérea, donde resulta necesario diferenciar correctamente entre amenazas potenciales y elementos naturales presentes en el entorno.

Para abordar este problema se han desarrollado y evaluado diferentes opciones. En primer lugar, se ha entrenado un modelo basado únicamente en imágenes RGB. Posteriormente, se ha propuesto una segunda variante que incorpora información adicional mediante el operador de Sobel, con el objetivo de resaltar los bordes de los objetos presentes en la imagen y reducir así la posible dependencia del modelo respecto al fondo. Finalmente, se ha incluido también una aproximación basada en *transfer-learning*, permitiendo comparar el rendimiento obtenido mediante entrenamiento desde cero con el uso de arquitecturas previamente entrenadas.

Por otra parte, los modelos desarrollados han sido evaluados también mediante imágenes tomadas manualmente fuera del conjunto de datos original, lo que ha permitido realizar una validación más cercana a un escenario real. Este aspecto resulta especialmente relevante, ya que las operaciones de vigilancia y detección de drones suelen llevarse a cabo en entornos abiertos donde la variabilidad de condiciones es elevada.

Los resultados obtenidos muestran que el modelo basado únicamente en RGB alcanza valores de *accuracy* elevados en el conjunto de datos empleado, lo que indica que la arquitectura propuesta es capaz de aprender correctamente las características necesarias para distinguir entre ambas clases. Además, el entrenamiento intensivo del modelo, junto con la aplicación de transformaciones de aumento de datos, ha demostrado ser una estrategia eficaz para mejorar la capacidad de generalización del sistema.

En cuanto al uso del operador de Sobel, los resultados obtenidos presentan un comportamiento interesante. Aunque el modelo que incorpora esta información obtiene resultados ligeramente inferiores en el conjunto de datos utilizado durante el entrenamiento y la evaluación, su comportamiento en pruebas realizadas con imágenes capturadas en situaciones reales muestra una mejora respecto al modelo RGB. Este hecho sugiere que el modelo entrenado únicamente con imágenes RGB puede haber aprendido en cierta medida patrones asociados al fondo de las imágenes del conjunto de datos, mientras que la incorporación de información de bordes contribuye a centrar la atención en las características estructurales del objeto.

Por su parte, el modelo basado en *transfer-learning* obtuvo excelentes resultados durante el entrenamiento sobre el *dataset*, pero mostró un rendimiento inferior a los otros dos al evaluarse con

imágenes externas. Este comportamiento sugiere que las características aprendidas durante el preentrenamiento no se adaptan completamente al dominio específico del problema abordado.

El uso de técnicas de interpretabilidad como Grad-CAM ha permitido analizar el comportamiento interno de los modelos entrenados. Los mapas de activación generados han resultado especialmente útiles para comprender qué regiones de la imagen influyen en la predicción del modelo, permitiendo detectar en algunos casos la dependencia respecto a elementos del fondo y facilitando la interpretación de los resultados obtenidos.

En conjunto, los resultados obtenidos permiten afirmar que el objetivo principal del trabajo se ha cumplido, diseñando una CNN propia desde cero, con el fin de diferenciar entre drones y pájaros. Además, se alcanzó con resultados satisfactorios, con valores de *accuracy* y *f1-score* por encima del 95% habituales en la literatura sobre CNN en este ámbito y una disminución hasta en torno al 87% de *accuracy* en el caso de imágenes reales ajenas al *dataset* en el modelo con Sobel. Esta disminución es predecible debido a que el cambio de dominio (*domain shift*) entre los datos de entrenamiento y los de evaluación puede degradar significativamente el rendimiento de los modelos, incluso en redes profundas [55].

Otro aspecto relevante a considerar durante el desarrollo del trabajo han sido las limitaciones técnicas encontradas durante el entrenamiento de los modelos. En particular, el proceso se ha llevado a cabo utilizando una GPU NVIDIA RTX 3060 con 12 GB de memoria, lo que condicionó el tamaño máximo de las imágenes utilizadas durante el entrenamiento. Como consecuencia, las imágenes del conjunto de datos fueron redimensionadas a una resolución de 448×448 píxeles. Aunque este tamaño ha permitido entrenar los modelos de forma eficiente, también introduce ciertas limitaciones en escenarios donde los drones aparecen a gran distancia, ya que en estos casos el objeto ocupa un número reducido de píxeles dentro de la imagen, dificultando su correcta identificación.

Asimismo, otra de las limitaciones observadas está relacionada con la influencia del fondo de las imágenes. Tal y como se ha podido comprobar mediante el análisis realizado con Grad-CAM, en algunos casos los modelos tienden a fijarse parcialmente en características del entorno en lugar de centrarse exclusivamente en el objeto de interés. Este comportamiento puede explicarse en parte por las características del conjunto de datos empleado, donde determinados tipos de fondos pueden aparecer con mayor frecuencia asociados a una de las clases. Aunque la introducción del operador de Sobel ha permitido mitigar parcialmente este efecto en las pruebas realizadas con imágenes externas, sigue siendo un aspecto susceptible de mejora.

5.2 Líneas futuras

Los resultados obtenidos en este trabajo establecen una base sólida para continuar investigando y mejorando los sistemas de detección automática de drones mediante técnicas de *deep learning*. A partir de las limitaciones identificadas durante el desarrollo del proyecto, se pueden plantear diversas líneas futuras de investigación orientadas a mejorar el rendimiento y la aplicabilidad del sistema.

- Ampliación del conjunto de datos

Una posible mejora consistiría en ampliar el *dataset* utilizado durante el entrenamiento, incorporando un mayor número de imágenes y una mayor variedad de escenarios. Esto permitiría reducir posibles sesgos asociados a determinados tipos de fondo y mejorar la capacidad de generalización del modelo ante situaciones no presentes en el conjunto de entrenamiento.

- Uso de imágenes de mayor resolución

Debido a las limitaciones de hardware durante el entrenamiento, las imágenes utilizadas fueron redimensionadas a 448×448 píxeles. En trabajos futuros podría explorarse el uso de resoluciones

superiores mediante el empleo de hardware más potente. Esto permitiría mejorar la detección de drones que aparecen a gran distancia o con un tamaño reducido dentro de la imagen.

- Ampliación del número de clases a detectar

El sistema desarrollado en este trabajo se centra en la clasificación binaria entre drones y aves. Sin embargo, en escenarios reales pueden aparecer otros objetos en el espacio aéreo, como aviones o helicópteros. Además, este trabajo tampoco ha sido entrenado con drones de ala fija, cada vez más utilizados en conflictos actuales. Por ello, una posible extensión del sistema consistiría en ampliar el número de clases consideradas, permitiendo una clasificación más completa de los objetos presentes en el entorno.

- Evaluación en escenarios más complejos

Otra línea futura interesante consistiría en evaluar el comportamiento del modelo en entornos más complejos, especialmente en contextos urbanos donde existen múltiples elementos visuales que pueden influir en la detección. Este tipo de pruebas permitiría analizar con mayor profundidad la robustez del sistema frente a fondos variados y condiciones visuales más exigentes.

- Aplicación en sistemas de detección en tiempo real

Finalmente, sería interesante explorar la integración del modelo en sistemas de monitorización en tiempo real, por ejemplo, mediante el análisis de secuencias de vídeo capturadas por cámaras de vigilancia o sensores ópticos. Esto permitiría evaluar la viabilidad del sistema en escenarios operativos reales.

En conjunto, las posibles mejoras planteadas permitirían aumentar la robustez, precisión y capacidad de adaptación del sistema desarrollado. La ampliación del conjunto de datos, el uso de resoluciones de imagen superiores o la evaluación en entornos más complejos contribuirían a acercar este tipo de soluciones a aplicaciones reales. De este modo, el trabajo presentado constituye un punto de partida sólido para el desarrollo de herramientas basadas en *deep learning* orientadas a la detección automática de drones en distintos escenarios operativos.

6 BIBLIOGRAFÍA

[1] A. Piñeiro Sánchez, «ARMADA 2050,» Armada, 2024.

[2] J. M. N. García, «defensa.com,» 4 Marzo 2026. [En línea]. Available: https://www.defensa.com/africa-asia-pacifico/lucas-dron-kamikaze-estadounidense-basado-shahed-irani-usado. [Último acceso: 6 Marzo 2026].

[3] R. Austin, Unmanned aircraft systems: UAVS Design, Development and Deployment, Wiley, 2010.

[4] G. Toribio, «cursosteledeteccion.com,» 9 Abril 2019. [En línea]. Available: https://www.cursosteledeteccion.com/drones-actualidad/. [Último acceso: 2 Marzo 2026].

[5] I. Guvenc , O. Ozdemir, Y. Yapici, H. Mehrpouyan y D. Matolak, «Detection, Localization, and Tracking of Unauthorized UAS and Jammers,» IEEE Communications Magazine, 2018.

[6] S. Morales Morales, «La segunda ola drónica,» 2023.

[7] A. Coluccia y A. Fascista, «Drone-vs-Bird Detection Challenge at IEEE AVSS2019,» 2021.

[8] D. Kunertova, «Drones have boots: Learning from Russia's war in Ukraine,» 4 Octubre 2023. [En línea]. Available: https://www.tandfonline.com/doi/full/10.1080/13523260.2023.2262792. [Último acceso: 19 Febrero 2026].

[9] Wikipedia, «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/HESA_Shahed_136.

[10] A. Holland Michel, «COUNTER-DRONE SYSTEMS,» 2018.

[11] S. Gazdag, T. Möller, T. Filep, A. Keszler y A. L. Majdik, «Detection and tracking of MAVs using a LiDAR with rosette scanning pattern,» 2024.

[12] «TERMA,» [En línea]. Available: https://www.terma.com/products/radars/3d-drone-. [Último acceso: 20 Febrero 2026].

[13] «Additional protection through visual detection. AARTOS Camera,» AARTOS, [En línea]. Available: https://drone-detection-system.com/sensor-types-overview/detection-camera/. [Último acceso: 18 Febrero 2026].

[14] R. Szeliski, «Computer Vision: Algorithms and Applications 2nd Edition,» Huree University of ICT, Ulaanbaatar, 2022.

[15] R. E. Woods y R. C. Gonzalez, Digital Image Processing 4ª Edición, New York: Pearson, 2018.

[16] P. Rosero-Montalvo, «ResearchGate.net,» Febrero 2017. [En línea]. Available: https://www.researchgate.net/figure/Figura-1-Fases-del-proceso-de-funcionamiento-de-la-vision-artificial-al-determinar-una_fig1_315682564. [Último acceso: 10 Marzo 2026].

[17] G. Dougherty, «Digital Image Processing for Medical Applications,» Cambridge, 2009.

[18] R. Alvarez, «defensa.com,» 28 Julio 2024. [En línea]. Available: https://www.defensa.com/industria/inteligencia-artificial-defensa-deteccion-identificacion2. [Último acceso: 24 Febrero 2026].

[19] D. A. Forsyth y J. Ponce, COMPUTER VISION A MODERN APPROACH 2nd Edition, Pearson, 2012.

[20] Y. LeCun, Y. Bengio y G. Hinton, «Deep learning,» *Nature,* vol. 521, nº 7553, pp. 436-444, 2015.

[21] P. Viola y M. Jones, «Rapid object detection using a boosted cascade of simple features,» de *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Kauai, HI, 2001.

[22] W. Hu, T. Tan, L. Wang y S. Maybank, «A survey on visual surveillance of object motion and behaviors,» *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 34, nº 3, pp. 334-352, 2004.

[23] I. Goodfellow, Y. Bengio y A. Courville, Deep Learning, Cambridge, MA: MIT Press, 2016.

[24] D. Calvo, «Diego Calvo,» 20 Julio 2017. [En línea]. Available: https://www.diegocalvo.es/red-neuronal-convolucional/. [Último acceso: 1 Marzo 2026].

[25] A. Krizhevsky, I. Sutskever y G. Hinton, «ImageNet classification with deep convolutional neural networks,» de *Advances in Neural Information Processing Systems*, Lake Tahoe, 2012.

[26] M. D. Zeiler y R. Fergus, «Visualizing and understanding convolutional networks,» de *European Conference on Computer Vision (ECCV)*, Zurich, 2014.

[27] D. G. Lowe, «Distinctive Image Features from Scale-Invariant Keypoints,» *International Journal of Computer Vision,* vol. 60, pp. 91-110, 2004.

[28] N. Dalal y B. Triggs, «Histograms of oriented gradients for human detection,» de *IEEE*, San Diego, CA, USA, 2005.

[29] F. Chollet, Deep Learning with Python, Shelter Island, NY: Manning Publications, 2017.

[30] R. Girshick, J. Donahue, T. Darrell y J. Malik, «Rich feature hierarchies for accurate object detection and semantic segmentation,» UC Berkeley, Berkeley, CA, 2014.

[31] S. Ren, K. He, R. Girshick y J. Sun, «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,» de *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu y A. C. Berg, «SSD: Single Shot MultiBox Detector,» de *European Conference on Computer Vision (ECCV)*, Amsterdam, 2016.

[33] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» de *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[34] A. Bochkovskiy, C.-Y. Wang y H.-Y. M. Liao, «YOLOv4: Optimal speed and accuracy of object detection,» 2020.

[35] G. Jocher, «Ultralytics,» 12 Noviembre 2023. [En línea]. Available: https://docs.ultralytics.com/models/yolov8/. [Último acceso: 3 Marzo 2026].

[36] A. Dosovitskiy, L. Beyer y A. Kolesnikov, «An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,» de *International Conference on Learning Representations (ICLR)*, Vienna, 2021.

[37] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles y H. Jégou, «Training data-efficient image transformers & distillation through attention,» de *International Conference on Machine Learning (ICML)*, Virtual Event, 2021.

[38] Z. Liu, Y. Lin y Y. Cao, «Swin Transformer: Hierarchical Vision Transformer using Shifted Windows,» de *IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, 2021.

[39] M. Rodrigo, C. Cuevas y N. Garcia, «Comprehensive comparison between vision transformers and convolutional neural networks for face recognition tasks,» 2024.

[40] J. Jakubik, M. Vössing, N. Kühl, J. Walk y G. Satzger, «Data-Centric Artificial Intelligence,» *Business & Information Systems Engineering,* 2022.

[41] B. C. Russell, A. Torralba, K. P. Murphy y W. Freeman, «LabelMe: a database and web-based tool for image annotation,» *International Journal of Computer Vision (IJCV),* vol. 77, pp. 157-173, 2008.

[42] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár y C. L. Zitnick, «Microsoft COCO: Common objects in context,» de *European Conference on Computer Vision (ECCV)*, Zurich, 2014.

[43] A. Torralba y A. A. Efros, «Unbiased look at dataset bias,» de *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, CO, 2011.

[44] J. Yosinski, J. Clune, Y. Bengio y H. Lipson, «How transferable are features in deep neural networks?,» de *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[45] H. M. Communiti & Conversation, «Facebook,» [En línea].

[46] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto y H. Adam, «MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,» Google Inc., 2017.

[47] J. Canales Luna, «DataCamp Blog,» 28 Enero 2025. [En línea]. Available: https://www.datacamp.com/es/blog/rectified-linear-unit-relu. [Último acceso: 10 Febrero 2026].

[48] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov y D. Erhan, «Going Deeper with Convolutions,» de *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[49] M. Lin, Q. Chen y S. Yan, «Network In Network,» 2013.

[50] G. Huang, Z. Liu, L. Van Der Maaten y K. Q. Weinberger, «Densely connected convolutional networks,» de *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[51] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan y S. Belongie, «Feature pyramid networks for object detection,» de *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[52] J. T. Springenberg, A. Dosovitskiy, T. Brox y M. Riedmiller, «Striving for simplicity: The all convolutional net,» de *International Conference on Learning Representations (ICLR)*, 2015.

[53] K. He, X. Zhang, S. Ren y J. Sun, «Deep residual learning for image recognition,» de *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[54] C. Shorten y T. M. Khoshgoftaar, «A survey on Image Data Augmentation for Deep Learning,» *Journal of Big Data,* vol. 6, nº 60, 2019.

[55] M. Wang y W. Deng, «Deep Visual Domain Adaptation: A Survey,» *Neurocomputing,* vol. 312, nº 135-153, 2018.

ANEXO I: IMPLICACIONES SOCIALES, Y/O ECONÓMICAS, Y/O AMBIENTALES

El desarrollo del trabajo se enmarca dentro del ámbito de la visión artificial aplicada a la clasificación de drones, una tecnología con implicaciones relevantes tanto a nivel social como económico y ambiental.

En el plano social, la mejora de los sistemas de detección e identificación de objetos aéreos contribuye directamente al aumento de la seguridad en entornos civiles y militares, especialmente en escenarios donde la presencia de drones puede suponer un riesgo.

Desde el punto de vista económico, el uso de modelos basados en aprendizaje profundo permite automatizar tareas de vigilancia que, de otro modo, requerirían una supervisión humana constante. Esto puede traducirse en una reducción de costes operativos a medio y largo plazo, aunque también implica una inversión inicial en recursos computacionales y desarrollo tecnológico.

En cuanto al impacto ambiental, es importante considerar el coste energético asociado al entrenamiento de modelos de *deep learning*. El uso de hardware como GPUs para el entrenamiento de redes neuronales supone un consumo energético significativo, especialmente cuando se realizan múltiples pruebas y ajustes del modelo. En este trabajo, se ha buscado limitar dicho impacto mediante un enfoque controlado del entrenamiento, evitando iteraciones innecesarias y optimizando los recursos disponibles.

ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

El desarrollo de sistemas automáticos de clasificación de imágenes, como el propuesto en este trabajo, plantea diversas cuestiones éticas que deben ser tenidas en cuenta. En primer lugar, el uso de tecnologías de visión artificial en el ámbito de la vigilancia puede generar preocupaciones relacionadas con la privacidad y el control del espacio aéreo, especialmente si estas herramientas se aplican en entornos civiles.

Por otro lado, un sistema que clasifique incorrectamente un objeto como dron o ave podría dar lugar a decisiones erróneas, con posibles consecuencias en contextos de seguridad. Por ello, resulta fundamental entender las limitaciones del modelo y evitar una confianza excesiva en sistemas automáticos sin supervisión.

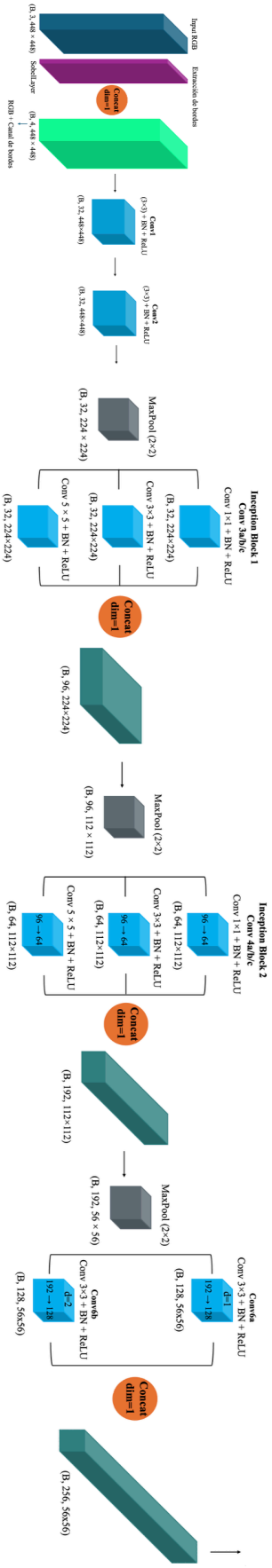
Asimismo, el uso de estas tecnologías en contextos militares, como se menciona en la motivación del trabajo, abre el debate sobre su posible aplicación en escenarios de conflicto. En este sentido, se considera importante que el desarrollo de tecnologías de inteligencia artificial vaya acompañado de un uso responsable y consciente de sus posibles impactos, tanto a nivel social como ético.

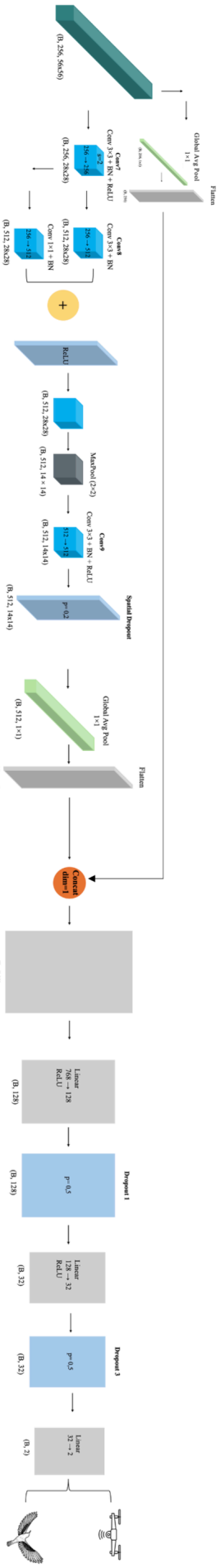
ANEXO III: FUENTES EMPLEADAS EN LA ELABORACIÓN DEL *DATASET*

Listado de las fuentes empleadas en la elaboración del *dataset* :

https://spdx.org/licenses/MIT.html
https://www.kaggle.com/datasets/dasmehdixtr/drone-dataset-uav?select=drone_dataset_yolo
https://universe.roboflow.com/ahmedmohsen/drone-detection-new-peksv
https://www.kaggle.com/datasets/stealthknight/bird-vs-drone
https://www.kaggle.com/datasets/nelyg8002000/birds-flying/code
https://portalinvestigacion.uniovi.es/documentos/67f62fbbc9d0c3013599a453 (iberbirds)
https://www.kaggle.com/datasets/harshwalia/birds-vs-drone-dataset
https://www.kaggle.com/datasets/cybersimar08/drone-detection
https://www.kaggle.com/datasets/imbikramsaha/drone-bird-classification
https://www.kaggle.com/datasets/muki2003/yolo-drone-detection-dataset
https://www.kaggle.com/datasets/sshikamaru/drone-yolo-detection
https://www.kaggle.com/datasets/sonainjamil/malicious-drones
https://www.kaggle.com/datasets/sarahalrowaily/birds-vs-drones-image-classification
https://www.kaggle.com/datasets/nelyg8002000/uav-detection-dataset-images
https://datasetninja.com/drone-dataset-uav
https://www.kaggle.com/datasets/mcagriaksoy/amateur-unmanned-air-vehicle-detection-dataset
https://www.kaggle.com/datasets/hussein1234/drone-uav-bird
https://www.kaggle.com/datasets/dasmehdixtr/drone-dataset-uav
https://www.kaggle.com/datasets/learnopencvblog/drones
https://www.kaggle.com/datasets/laliev/drones
https://github.com/Maciullo/DroneDetectionDataset
https://universe.roboflow.com/customyolo-k3lvg/drones-rom1a
https://datasetninja.com/accurate-drone

ANEXO IV: ESQUEMA DE BLOQUES CNN




(B, 256, 56×56)
Global Avg Pool
1×1
Flatten
Conv7
Conv 3×3 + BN + ReLU
256 → 256
(B, 256, 28×28)
Conv8
Conv 3×3 + BN
256 → 512
(B, 512, 28×28)
Conv 1×1 + BN
256 → 512
(B, 512, 28×28)
+
ReLU
(B, 512, 28×28)
MaxPool (2×2)
(B, 512, 14 × 14)
Conv9
Conv 3×3 + BN + ReLU
512 → 512
(B, 512, 14×14)
Spatial Dropout
p= 0.2
(B, 512, 14×14)
Global Avg Pool
1×1
(B, 512, 1×1)
Flatten
(B, 512)
Concat
dim=1
(B, 768)
Linear
768 → 128
ReLU
(B, 128)
Dropout 1
p= 0.5
(B, 128)
Linear
128 → 32
ReLU
(B, 32)
Dropout 3
p= 0.5
(B, 32)
Linear
32 → 2
(B, 2)

ANEXO V: TABLAS RESUMEN TÉCNICO DE PARÁMETROS DE AMBOS MODELOS

Resumen del modelo de tres canales (RGB):

```
----------------------------------------------------------------
    Layer (type)               Output Shape         Param #
================================================================
       Conv2d-1         [-1, 32, 448, 448]             896
    BatchNorm2d-2         [-1, 32, 448, 448]              64
       Conv2d-3         [-1, 32, 448, 448]           9,248
    BatchNorm2d-4         [-1, 32, 448, 448]              64
     MaxPool2d-5         [-1, 32, 224, 224]               0
       Conv2d-6         [-1, 32, 224, 224]           1,056
    BatchNorm2d-7         [-1, 32, 224, 224]              64
       Conv2d-8         [-1, 32, 224, 224]           9,248
    BatchNorm2d-9         [-1, 32, 224, 224]              64
      Conv2d-10         [-1, 32, 224, 224]          25,632
   BatchNorm2d-11         [-1, 32, 224, 224]              64
    MaxPool2d-12         [-1, 96, 112, 112]               0
      Conv2d-13         [-1, 64, 112, 112]           6,208
   BatchNorm2d-14         [-1, 64, 112, 112]             128
      Conv2d-15         [-1, 64, 112, 112]          55,360
   BatchNorm2d-16         [-1, 64, 112, 112]             128
      Conv2d-17         [-1, 64, 112, 112]         153,664
   BatchNorm2d-18         [-1, 64, 112, 112]             128
    MaxPool2d-19         [-1, 192, 56, 56]               0
      Conv2d-20         [-1, 128, 56, 56]          221,312
   BatchNorm2d-21         [-1, 128, 56, 56]             256
      Conv2d-22         [-1, 128, 56, 56]          221,312
   BatchNorm2d-23         [-1, 128, 56, 56]             256
AdaptiveAvgPool2d-24           [-1, 256, 1, 1]               0
      Conv2d-25         [-1, 256, 28, 28]          590,080
   BatchNorm2d-26         [-1, 256, 28, 28]             512
      Conv2d-27         [-1, 512, 28, 28]       1,180,160
   BatchNorm2d-28         [-1, 512, 28, 28]           1,024
      Conv2d-29         [-1, 512, 28, 28]         131,584
   BatchNorm2d-30         [-1, 512, 28, 28]           1,024
    MaxPool2d-31         [-1, 512, 14, 14]               0
      Conv2d-32         [-1, 512, 14, 14]       2,359,808
   BatchNorm2d-33         [-1, 512, 14, 14]           1,024
    Dropout2d-34         [-1, 512, 14, 14]               0
AdaptiveAvgPool2d-35           [-1, 512, 1, 1]               0
      Linear-36                 [-1, 128]          98,432
     Dropout-37                 [-1, 128]               0
      Linear-38                  [-1, 32]           4,128
     Dropout-39                  [-1, 32]               0
      Linear-40                   [-1, 2]              66
================================================================
                  Total params: 5,072,994
              Trainable params: 5,072,994
                Non-trainable params: 0
----------------------------------------------------------------
                  Input size (MB): 2.30
          Forward/backward pass size (MB): 362.91
                  Params size (MB): 19.35
             Estimated Total Size (MB): 384.56
----------------------------------------------------------------
```

Resumen del modelo de cuatro canales (RGB + bordes):

```
----------------------------------------------------------------
     Layer (type)               Output Shape         Param #
================================================================
     SobelLayer-1          [-1, 1, 448, 448]               0
       Conv2d-2         [-1, 32, 448, 448]           1,184
    BatchNorm2d-3         [-1, 32, 448, 448]              64
       Conv2d-4         [-1, 32, 448, 448]           9,248
    BatchNorm2d-5         [-1, 32, 448, 448]              64
     MaxPool2d-6         [-1, 32, 224, 224]               0
       Conv2d-7         [-1, 32, 224, 224]           1,056
    BatchNorm2d-8         [-1, 32, 224, 224]              64
       Conv2d-9         [-1, 32, 224, 224]           9,248
   BatchNorm2d-10         [-1, 32, 224, 224]              64
      Conv2d-11         [-1, 32, 224, 224]          25,632
   BatchNorm2d-12         [-1, 32, 224, 224]              64
    MaxPool2d-13         [-1, 96, 112, 112]               0
      Conv2d-14         [-1, 64, 112, 112]           6,208
   BatchNorm2d-15         [-1, 64, 112, 112]             128
      Conv2d-16         [-1, 64, 112, 112]          55,360
   BatchNorm2d-17         [-1, 64, 112, 112]             128
      Conv2d-18         [-1, 64, 112, 112]         153,664
   BatchNorm2d-19         [-1, 64, 112, 112]             128
    MaxPool2d-20         [-1, 192, 56, 56]               0
      Conv2d-21         [-1, 128, 56, 56]         221,312
   BatchNorm2d-22         [-1, 128, 56, 56]             256
      Conv2d-23         [-1, 128, 56, 56]         221,312
   BatchNorm2d-24         [-1, 128, 56, 56]             256
AdaptiveAvgPool2d-25         [-1, 256, 1, 1]               0
      Conv2d-26         [-1, 256, 28, 28]         590,080
   BatchNorm2d-27         [-1, 256, 28, 28]             512
      Conv2d-28         [-1, 512, 28, 28]       1,180,160
   BatchNorm2d-29         [-1, 512, 28, 28]           1,024
      Conv2d-30         [-1, 512, 28, 28]         131,584
   BatchNorm2d-31         [-1, 512, 28, 28]           1,024
    MaxPool2d-32         [-1, 512, 14, 14]               0
      Conv2d-33         [-1, 512, 14, 14]       2,359,808
   BatchNorm2d-34         [-1, 512, 14, 14]           1,024
    Dropout2d-35         [-1, 512, 14, 14]               0
AdaptiveAvgPool2d-36         [-1, 512, 1, 1]               0
     Linear-37                  [-1, 128]          98,432
    Dropout-38                  [-1, 128]               0
     Linear-39                   [-1, 32]           4,128
    Dropout-40                   [-1, 32]               0
     Linear-41                    [-1, 2]              66
================================================================
                  Total params: 5,073,282
              Trainable params: 5,073,282
          Non-trainable params: 0
----------------------------------------------------------------
                  Input size (MB): 2.30
        Forward/backward pass size (MB): 364.45
                  Params size (MB): 19.35
           Estimated Total Size (MB): 386.10
----------------------------------------------------------------
```

ANEXO VI: USO DE IAG

Durante el desarrollo del trabajo se utilizaron herramientas de apoyo basadas en inteligencia artificial, principalmente para facilitar tareas de programación y búsqueda de posibles soluciones. En concreto, se emplearon para obtener ejemplos de código, plantear distintas formas de implementar algunas partes del sistema y como ayuda inicial en fases de prueba.

También se usaron de forma puntual como apoyo en la redacción de algunos fragmentos de la memoria, sobre todo para mejorar la forma de expresar ciertas ideas o hacer el texto más claro.

Las utilizadas fueron ChatGPT, Google Gemini, Microsoft Copilot y Google NotebookLM, con el fin de comparar diferentes enfoques y versiones de trabajo, adaptando el fin de la consulta realizada a la fuente más óptima para cada actividad a realizar.

En cualquier caso, todas las aportaciones obtenidas mediante estas herramientas fueron revisadas, adaptadas y validadas antes de incorporarlas al trabajo, asegurando que el resultado final fuese coherente con los objetivos planteados y reflejase el trabajo propio realizado.