



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Detección automatizada de objetos en entornos marinos
mediante visión termográfica y técnicas de Deep Learning*

Grado en Ingeniería Mecánica

ALUMNO: María Vigo Aleu

DIRECTORES: Iván Garrido González
Leticia Pérez Rial

CURSO ACADÉMICO: 2025-2026

Universida_{de}Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Detección automatizada de objetos en entornos marinos
mediante visión termográfica y técnicas de Deep Learning*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

UniversidadeVigo

RESUMEN

En el ámbito marítimo, la detección temprana de objetos resulta fundamental para garantizar la seguridad marítima y apoyar las operaciones de vigilancia y defensa, especialmente en escenarios de baja visibilidad donde los sistemas ópticos tradicionales presentan limitaciones.

El presente Trabajo Fin de Grado da continuidad a un estudio previo desarrollado en el CUD, centrado en la comparación entre cámaras termográficas y convencionales en condiciones climáticas adversas. Con el objetivo de avanzar en seguridad y defensa naval, este trabajo incorpora la automatización inteligente del proceso de detección de objetos de interés en entornos marítimos a partir de imágenes térmicas.

Para ello, se han empleado dos modelos de *Deep Learning* basados en redes neuronales convolucionales, de características complementarias: YOLO, enfocado en la detección rápida de la ubicación de objetos, y Mask R-CNN, centrado en la delimitación precisa de los contornos. Los objetos se clasifican en función de su tamaño y distancia en cuatro categorías: grande-cerca, grande-lejos, pequeño-cerca y pequeño-lejos.

Los resultados obtenidos demuestran la viabilidad del enfoque propuesto, destacando la robustez del modelo YOLO, con un porcentaje de acierto del $80,75\% \pm 9,95\%$. Este trabajo refuerza el potencial de la visión termográfica combinada con inteligencia artificial en entornos marítimos reales.

PALABRAS CLAVE

Automatización, Cámaras termográficas, *Deep Learning*, Redes neuronales convolucionales, Seguridad marítima

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a todas aquellas personas que han contribuido, de una forma u otra, a la realización del presente Trabajo Fin de Grado.

En primer lugar, a mis tutores, por su orientación, disponibilidad y apoyo a lo largo de todo el desarrollo del trabajo. Su experiencia y sus recomendaciones han sido fundamentales para enfocar correctamente el trabajo y superar las distintas dificultades encontradas durante el proceso.

A mis compañeros de promoción, por el camino compartido, el apoyo diario y el esfuerzo conjunto que han hecho de esta etapa una experiencia única. Del mismo modo, a mis amigas, aunque separadas por la distancia, por estar siempre presentes y por su apoyo constante a lo largo de estos años.

Por último, a mi familia, por ser el pilar fundamental que me ha permitido llegar hasta aquí, agradezco profundamente el apoyo, la confianza y los valores que me han transmitido. Dedico de manera especial este trabajo a la memoria de mi padre, cuyo recuerdo es una fuente constante de motivación para alcanzar cada uno de mis objetivos.

CONTENIDO

Contenido	1
Índice de Figuras	3
Índice de Tablas.....	8
1 Introducción y objetivos	9
1.1 Contexto	9
1.2 Objetivos	10
1.3 Metodología	10
1.4 Estructura	10
2 Estado del arte	12
2.1 Seguridad marítima a nivel global	12
2.2 Seguridad y vigilancia marítima en la Armada.....	13
2.2.1 Amenazas y riesgos	13
2.2.2 Marco estratégico de seguridad nacional y marítima	14
2.2.3 Competencias.....	14
2.3 Sensores y sistemas empleados en entornos marinos	15
2.3.1 Sistemas radar.....	15
2.3.2 Sistemas de identificación automática de buques (AIS).....	16
2.3.3 Sistemas de seguridad y socorro marítimo (GMDSS).....	17
2.3.4 Sistemas ópticos convencionales	18
2.3.4.1 Cámaras en el espectro visible	18
2.3.4.2 Cámaras en el espectro infrarrojo.....	18
2.3.5 Limitaciones	19
2.4 Termografía infrarroja.....	20
2.4.1 Generalidades	20
2.4.2 Principios físicos.....	21
2.4.3 Modos de operación.....	22
2.4.4 Aplicación en el ámbito marítimo	23
2.5 Inteligencia Artificial en la detección de objetos.....	24
2.5.1 Introducción a la IA	24
2.5.2 Aprendizaje automático (Machine Learning).....	25
2.5.3 Aprendizaje profundo (Deep Learning).....	26
2.6 Redes Neuronales Convolucionales.....	27
2.6.1 Fundamentos.....	27
2.6.2 Aplicación en imágenes y detección de objetos	28

3 Desarrollo del TFG.....	31
3.1 Área de estudio.....	31
3.1.1 Base fija	32
3.1.2 Base móvil	33
3.1.3 Objetos de interés	34
3.2 Equipos empleados para la toma de imágenes	35
3.2.1 Cámara termográfica FLIR E-85	35
3.2.2 Cámara termográfica FLIR One Edge	37
3.3 Procesamiento de imágenes	38
3.3.1 Clasificación y etiquetado.....	38
3.3.2 Ejecución de los modelos Deep Learning	41
3.3.2.1 YOLO	42
3.3.2.2 Mask R-CNN.....	45
4 Resultados y Discusión	49
4.1 Resultados en el Escenario 1	49
4.2 Resultados en el Escenario 2.....	58
4.3 Resultados en el Escenario 3.....	65
4.4 Análisis y comparaciones.....	71
5 Conclusiones y líneas futuras	75
5.1 Cumplimiento de los objetivos	75
5.2 Líneas futuras de investigación.....	76
6 Bibliografía.....	77
Anexo I: Implicaciones Sociales, y/o Económicas, y/o Ambientales	81
Anexo II: Reflexiones Éticas y Sociales	83
Anexo III: Código YOLO v11 y v26	84
Anexo IV: Código Mask R-CNN.....	88
Anexo V: Declaración de uso de Inteligencia Artificial	98

ÍNDICE DE FIGURAS

Figura 2.1 El comercio marítimo mundial [6].....	13
Figura 2.2 Competencias y principales operaciones de la Armada en el ámbito de la seguridad marítima (Elaboración propia) [12]	15
Figura 2.3 Funcionamiento básico de un sistema radar de navegación [14].....	16
Figura 2.4 Funcionamiento detallado del sistema AIS [17]	17
Figura 2.5 Funcionamiento del sistema GMDSS [19]	17
Figura 2.6 Esquema de cámara convencional para espectro visible [20].....	18
Figura 2.7 Imagen de un cuadro eléctrico en el espectro infrarrojo tomada con la cámara termográfica modelo IR-282 [23]	19
Figura 2.8 Ejemplo de imagen termográfica [25]	20
Figura 2.9 Espectro electromagnético [27]	21
Figura 2.10 Ley de Planck para cuerpos a diferentes temperaturas [28].....	22
Figura 2.11 Comparativa de imágenes del espectro visible e infrarrojo de una cámara FLIR en la mar [30].....	23
Figura 2.12 Categorías y aplicaciones actuales más comunes de la IA [34].....	24
Figura 2.13 Procesamiento de imágenes para la extracción de objetos [35].....	25
Figura 2.14 Algoritmo de <i>Deep Learning</i> basado en una red neuronal [38].....	26
Figura 2.15 Esquema de funcionamiento de la CNN [41]	28
Figura 2.16 Arquitectura de red del modelo YOLO [43].....	28
Figura 2.17 Vista general de la arquitectura Mask R-CNN [44].....	29
Figura 2.18 Arquitectura de red neuronal convolucional con SSD [45]	29
Figura 2.19 Arquitectura de red neuronal U-Net [46].....	29
Figura 3.1 Localización de la Escuela Naval Militar en la Ría de Pontevedra (Coordenadas 42° 23' N, 008° 42'W) (Fuente: Google Earth).....	31
Figura 3.2 Vista general del área de estudio dentro del recinto de la Escuela Naval Militar (Fuente: Google Earth).....	32
Figura 3.3 Distribución de las bases fijas escogidas para la toma de imágenes (Fuente: Google Earth)	32
Figura 3.4 Vista general desde la zona B1 (Elaboración propia)	33
Figura 3.5 Vista general desde la zona B2 (Elaboración propia)	33
Figura 3.6 Vista general desde la zona B3 (Elaboración propia)	33
Figura 3.7 Lancha de instrucción A-121 'Guardiamarina Barrutia' (Fuente: Facebook 'Escuela Naval Militar')	34
Figura 3.8 Embarcación auxiliar Y-504 en la Escuela Naval Militar [50].....	34
Figura 3.9 Embarcaciones de gran tamaño en tránsito o fondeados en la ría (Elaboración propia)	34
Figura 3.10 Palangre a la deriva en la ría (Elaboración propia).....	35

Figura 3.11 Muñeco de simulación de hombre al agua en ejercicios de lanchas de instrucción (Elaboración propia)	35
Figura 3.12 Cámara termográfica FLIR E-85 [51].....	36
Figura 3.13 Comparación de imágenes térmica y visible realizadas con la FLIR E-85 (Elaboración propia)	36
Figura 3.14 Cámara termográfica FLIR One Edge [54].....	37
Figura 3.15 Comparación de imágenes térmica y visible realizadas con la FLIR One Edge (Elaboración propia)	38
Figura 3.16 Ejemplo de nomenclatura de imagen térmica (Elaboración propia).....	39
Figura 3.17 Primer paso de carga de imágenes en la página principal de la herramienta VGG (Elaboración propia)	40
Figura 3.18 Ejemplo del procedimiento de creación de atributos (Elaboración propia).....	40
Figura 3.19 Procedimiento de trazado de la región y opciones de clasificación (Elaboración propia)	41
Figura 3.20 Código del archivo .json exportado con las características del etiquetado de la imagen (Elaboración propia)	41
Figura 3.21 Esquema general del funcionamiento del modelo YOLO empleado para la detección de objetos (Elaboración propia).....	43
Figura 3.22 Esquema general del funcionamiento del modelo Mask R-CNN empleado para la detección de objetos (Elaboración propia).....	46
Figura 4.1 Resultados gráficos de las funciones de pérdida (<i>loss</i>) de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) en el Escenario 1 (Elaboración propia)	50
Figura 4.2 Resultado de las métricas de evaluación de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) en el Escenario 1 (Elaboración propia)	51
Figura 4.3 Matrices de confusión normalizadas de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) del Escenario 1 (Elaboración propia).....	52
Figura 4.4 Ejemplos de imágenes ‘ <i>LABEL</i> ’ (a la izquierda) y ‘ <i>PRED</i> ’ (a la derecha) del modelo YOLOv11 en el Escenario 1 (Elaboración propia).....	53
Figura 4.5 Ejemplos de imágenes ‘ <i>LABEL</i> ’ (a la izquierda) y ‘ <i>PRED</i> ’ (a la derecha) del modelo YOLOv26 en el Escenario 1 (Elaboración propia).....	53
Figura 4.6 Resultados de la fase de testeo del Escenario 1 del modelo YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) (Elaboración propia).....	54
Figura 4.7 Gráficas de pérdidas del modelo Mask R-CNN durante la fase de entrenamiento en el Escenario 1 (Elaboración propia).....	54
Figura 4.8 Gráficas de pérdidas asociadas a la <i>RPN</i> en el Escenario 1 (Elaboración propia).....	55
Figura 4.9 Métricas de evaluación del <i>bounding box</i> del conjunto de validación del Escenario 1 (Elaboración propia)	55
Figura 4.10 Métricas de evaluación de la segmentación del conjunto de validación del Escenario 1 (Elaboración propia)	56
Figura 4.11 Matriz de confusión normal (a la izquierda) y normalizada (a la derecha) del modelo Mask R-CNN en el Escenario 1 (Elaboración propia).....	56

Figura 4.12 Ejemplos de imágenes ‘ <i>LABEL</i> ’ (a la izquierda) y ‘ <i>PRED</i> ’ (a la derecha) del modelo Mask R-CNN en el Escenario 1 (Elaboración propia).....	57
Figura 4.13 Resultados de la fase de testeo del modelo Mask R-CNN en el Escenario 1 (Elaboración propia).....	57
Figura 4.14 Resultados gráficos de las funciones de pérdida (<i>loss</i>) de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) en el Escenario 2 (Elaboración propia)	58
Figura 4.15 Resultado de las métricas de evaluación de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) en el Escenario 2 (Elaboración propia)	59
Figura 4.16 Matrices de confusión normalizadas de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) del Escenario 2 (Elaboración propia)	59
Figura 4.17 Ejemplos de imágenes ‘ <i>LABEL</i> ’ (a la izquierda) y ‘ <i>PRED</i> ’ (a la derecha) del modelo YOLOv11 en el Escenario 2 (Elaboración propia).....	60
Figura 4.18 Ejemplos de imágenes ‘ <i>LABEL</i> ’ (a la izquierda) y ‘ <i>PRED</i> ’ (a la derecha) del modelo YOLOv26 en el Escenario 2 (Elaboración propia).....	60
Figura 4.19 Resultados de la fase de testeo del Escenario 2 del modelo YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) (Elaboración propia).....	61
Figura 4.20 Gráficas de pérdidas del modelo Mask R-CNN durante la fase de entrenamiento en el Escenario 2 (Elaboración propia).....	61
Figura 4.21 Gráficas de pérdidas asociadas a la <i>RPN</i> en el Escenario 2 (Elaboración propia).....	62
Figura 4.22 Métricas de evaluación del <i>bounding box</i> del conjunto de validación del Escenario 2 (Elaboración propia)	62
Figura 4.23 Métricas de evaluación de la segmentación del conjunto de validación del Escenario 2 (Elaboración propia)	63
Figura 4.24 Matriz de confusión normal (a la izquierda) y normalizada (a la derecha) del modelo Mask R-CNN en el Escenario 2 (Elaboración propia).....	63
Figura 4.25 Ejemplos de imágenes ‘ <i>LABEL</i> ’ (a la izquierda) y ‘ <i>PRED</i> ’ (a la derecha) del modelo Mask R-CNN en el Escenario 2 (Elaboración propia).....	64
Figura 4.26 Resultados de la fase de testeo del modelo Mask R-CNN en el Escenario 2 (Elaboración propia)	64
Figura 4.27 Resultados gráficos de las funciones de pérdida (<i>loss</i>) de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) en el Escenario 3 (Elaboración propia)	65
Figura 4.28 Resultado de las métricas de evaluación de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) en el Escenario 3 (Elaboración propia)	66
Figura 4.29 Matrices de confusión normalizadas de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) del Escenario 3 (Elaboración propia)	66
Figura 4.30 Ejemplos de imágenes ‘ <i>LABEL</i> ’ (a la izquierda) y ‘ <i>PRED</i> ’ (a la derecha) del modelo YOLOv11 en el Escenario 3 (Elaboración propia).....	67
Figura 4.31 Ejemplos de imágenes ‘ <i>LABEL</i> ’ (a la izquierda) y ‘ <i>PRED</i> ’ (a la derecha) del modelo YOLOv26 en el Escenario 3 (Elaboración propia).....	67
Figura 4.32 Resultados de la fase de testeo del Escenario 3 del modelo YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) (Elaboración propia).....	68

Figura 4.33 Gráficas de pérdidas del modelo Mask R-CNN durante la fase de entrenamiento en el Escenario 3 (Elaboración propia).....	68
Figura 4.34 Gráficas de pérdidas asociadas a la <i>RPN</i> en el Escenario 3 (Elaboración propia).....	68
Figura 4.35 Métricas de evaluación del <i>bounding box</i> del conjunto de validación del Escenario 3 (Elaboración propia)	69
Figura 4.36 Métricas de evaluación de la segmentación del conjunto de validación del Escenario 3 (Elaboración propia)	69
Figura 4.37 Matriz de confusión normal (a la izquierda) y normalizada (a la derecha) del modelo Mask R-CNN en el Escenario 3 (Elaboración propia).....	70
Figura 4.38 Ejemplos de imágenes ‘ <i>LABEL</i> ’ (a la izquierda) y ‘ <i>PRED</i> ’ (a la derecha) del modelo Mask R-CNN en el Escenario 3 (Elaboración propia).....	70
Figura 4.39 Resultados de la fase de testeo del modelo Mask R-CNN en el Escenario 3 (Elaboración propia)	71
Figura 4.40 Ejemplo de detecciones en la misma imagen tras el proceso de testeo de los tres modelos (Elaboración propia)	74
Figura 4.41 Comparativa de probabilidades de detección en imágenes de distintas fuentes mediante el modelo YOLO (Elaboración propia)	74
Figura A3.1 Entorno de ejecución del código YOLO en Google Colab (Elaboración propia).....	84
Figura A3.2 Montaje de Google Drive y comprobación del empleo de la GPU (Elaboración propia)	84
Figura A3.3 Función de llamada para conversión a formato específico YOLO (Elaboración propia)	85
Figura A3.4 Creación del dataset y extracción de archivos comprimidos (Elaboración propia)	85
Figura A3.5 Código ejecutado para la división automática en entrenamiento y validación (Elaboración propia)	85
Figura A3.6 Comando de instalación de la librería <i>Ultralytics</i> (Elaboración propia)	85
Figura A3.7 Comando y resultado de la creación del archivo <i>.yaml</i> (Elaboración propia)	86
Figura A3.8 Entrenamiento y resultados del proceso de entrenamiento (Elaboración propia).....	86
Figura A3.9 Comando para el testeo del modelo y resultado visual de las predicciones en las imágenes procesadas (Elaboración propia).....	87
Figura A4.1 Entorno de ejecución del código Mask R-CNN en Google Colab (Elaboración propia)	88
Figura A4.2 Montaje de Google Drive e instalación de librerías (Elaboración propia)	88
Figura A4.3 Recopilación de rutas e hiperparámetros para la configuración del modelo (Elaboración propia)	89
Figura A4.4 Procesamiento de anotaciones y registro de <i>datasets</i> (Elaboración propia)	89
Figura A4.5 Configuración inicial del modelo Mask R-CNN (Elaboración propia)	90
Figura A4.6 Implementación de <i>DefaultTrainer</i> para el entrenamiento del modelo (Elaboración propia)	90
Figura A4.7 Librerías de personalización del entrenador y preparación del <i>dataset</i> para el aprendizaje (Elaboración propia)	91

Figura A4.8 Configuración final del entrenamiento (Elaboración propia)	92
Figura A4.9 Establecimiento de las iteraciones de evaluación, inclusión de <i>hooks</i> e inicio del entrenamiento (Elaboración propia).....	92
Figura A4.10 Librerías de validación, reajuste del umbral de confianza y ejecución de la evaluación final del modelo en el conjunto de validación (Elaboración propia)	93
Figura A4.11 Generación de gráficos de métricas de validación y matrices de confusión (Elaboración propia)	94
Figura A4.12 Extracción y guardado de métricas <i>mAP</i> de los resultados de la evaluación (Elaboración propia)	95
Figura A4.13 Obtención metadatos de validación y selección de imágenes para la preparación del <i>Visualizer</i> (Elaboración propia)	95
Figura A4.14 Carga de imágenes, iteraciones y obtención de predicciones para su visualización (Elaboración propia)	96
Figura A4.15 Proceso de inferencia del modelo para imágenes de test no utilizadas (Elaboración propia)	97

ÍNDICE DE TABLAS

Tabla 2-1 Resumen de diferencias entre la termografía activa y pasiva (Elaboración propia).....	23
Tabla 2-2 Comparación de características de los modelos Mask R-CNN y YOLO (Elaboración propia)	30
Tabla 3-1 Características principales de la FLIR E-85 [53].....	36
Tabla 3-2 Características principales de la FLIR One Edge [55].....	37
Tabla 4-1 Resumen comparativo de las métricas en el conjunto de validación del Escenario 1 (Elaboración propia)	72
Tabla 4-2 Resumen comparativo de las métricas en el conjunto de validación del Escenario 2 (Elaboración propia)	73
Tabla 4-3 Resumen comparativo de las métricas en el conjunto de validación del Escenario 3 (Elaboración propia)	73

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Contexto

El medio marino constituye uno de los escenarios más complejos y exigentes para la observación, la vigilancia y la navegación, debido a su elevada variabilidad ambiental y a la influencia constante de factores meteorológicos y oceanográficos. Fenómenos como la niebla, la lluvia, el oleaje o la ausencia de luz natural limitan de forma significativa la capacidad de los sistemas de detección tradicionales, incrementando el riesgo de accidentes, colisiones y situaciones de peligro tanto para la navegación civil como militar. En este contexto, la identificación oportuna de objetos flotantes, desde grandes embarcaciones hasta desechos o personas en situación de riesgo, constituye un factor determinante para salvaguardar vidas, preservar activos navales y garantizar el cumplimiento de operaciones de vigilancia y defensa. Históricamente, estas tareas han recaído en sistemas como el radar, el AIS o las cámaras ópticas convencionales. No obstante, las inclemencias meteorológicas frecuentes comprometen gravemente la eficacia de los sistemas de imagen visible y radares convencionales, al saturar las señales con ruido ambiental y exigir una supervisión humana continua que resulta ineficiente a gran escala.

En operaciones marítimas modernas, las cámaras termográficas emergen como alternativa prometedora a estas limitaciones, al capturar emisiones infrarrojas emitidas por los objetos independientes de la iluminación visible. Estudios previos han puesto de manifiesto la superioridad de este tipo de sensores en escenarios de baja visibilidad o con condiciones desfavorables, así como la viabilidad del empleo de cámaras termográficas de bajo coste como complemento a sistemas de mayor gama. Aun así, en la mayoría de los casos el análisis manual de estas imágenes limita su despliegue práctico en operaciones reales de la Armada, donde se requiere procesar flujos de datos en tiempo real desde plataformas fijas o móviles.

En paralelo, durante los últimos años se ha producido un avance significativo en el campo de la inteligencia artificial, y en particular en las técnicas de *Deep Learning* o aprendizaje profundo, aplicadas al reconocimiento y procesamiento de voz e imágenes, así como al lenguaje natural y asistentes virtuales. Del mismo modo, las redes neuronales convolucionales han demostrado una gran capacidad para la detección automática de objetos en múltiples ámbitos, permitiendo identificar patrones complejos a partir de grandes cantidades de datos, realizando así tareas de reconocimiento de imágenes y voz de forma robusta y eficiente. La integración de estas técnicas con sensores termográficos abre nuevas posibilidades para la automatización inteligente de los sistemas de vigilancia marítima, reduciendo la carga del operador humano y aumentando la rapidez y fiabilidad en la toma de decisiones.

1.2 Objetivos

El presente Trabajo Fin de Grado tiene como objetivo el desarrollo y evaluación de un sistema de detección automatizada de objetos de interés en entornos marinos mediante el uso de imágenes termográficas y técnicas avanzadas de inteligencia artificial. Frente a los métodos tradicionales basados en la observación y análisis de imágenes, este trabajo plantea el aprendizaje y testeo de modelos de *Deep Learning*, basados en redes neuronales convolucionales, capaces de identificar y localizar de forma automática distintos objetos presentes en escenas marítimas a partir de información térmica.

Para ello, se emplearán cámaras termográficas de distinta gama, incluyendo una cámara de altas prestaciones y una alternativa de bajo coste, con el objetivo de evaluar la viabilidad de soluciones tecnológicamente accesibles desde el punto de vista económico. Las imágenes se adquirirán tanto desde plataformas fijas como móviles en el entorno de la Escuela Naval Militar y la ría de Pontevedra, proporcionando un escenario realista y de interés operativo. A partir de este conjunto de datos se aplicarán dos modelos de *Deep Learning* ampliamente utilizados en visión artificial: ‘YOLO’ y ‘Mask R-CNN’. El uso de ambos permitirá comparar sus prestaciones y analizar si la mayor rapidez de ‘YOLO’ compensa su menor precisión frente a la segmentación más detallada proporcionada por ‘Mask R-CNN’.

La motivación principal de este trabajo radica en su aplicación directa en el ámbito de la seguridad marítima y la defensa, áreas de especial relevancia para los futuros oficiales de la Armada. El desarrollo de sistemas de detección inteligentes, capaces de operar de forma eficaz en condiciones adversas y con un alto grado de automatización, constituye un paso fundamental hacia la modernización de los sistemas de vigilancia naval y la mejora de la seguridad en la mar.

1.3 Metodología

Para alcanzar los objetivos previamente planteados, se ha definido como entorno de estudio adecuado la ría de Pontevedra y las inmediaciones de la Escuela Naval Militar, en Marín, por tratarse de un escenario realista y de interés para aplicaciones de seguridad marítima.

A continuación, se describen de forma resumida los métodos seguidos para el desarrollo del presente trabajo:

- Adquisición de imágenes relacionadas con objetos de interés en el medio marino mediante una cámara de visión termográfica de alta gama (FLIR E-85) y otra de bajo coste (FLIR ONE EDGE), tomadas desde diferentes tipos de plataforma (tanto móviles en embarcaciones como fijas en tierra) y bajo condiciones ambientales favorables y desfavorables.
- Procesamiento de las imágenes obtenidas, mediante su recopilación, preparación y el etiquetado de los objetos de interés presentes. Las imágenes procesadas constituirán la base del conjunto de datos empleado en la posterior aplicación del código de los modelos propuestos.
- Aprendizaje (entrenamiento y validación) y testeo de los modelos de *Deep Learning* ‘YOLO’ y ‘MASK R-CNN’ basados en redes neuronales convolucionales, para conseguir la detección automatizada de los objetos marinos en las imágenes termográficas.
- Análisis y discusión de los resultados obtenidos mediante la comparativa de ambos modelos y evaluación de las posibilidades y limitaciones del sistema, así como posibles líneas de mejora y trabajos futuros.

1.4 Estructura

El trabajo estará dividido en cuatro capítulos resumidos en los siguientes puntos:

1. Estado del arte: se exponen los principales conceptos sobre los que trata el trabajo, relacionados con la seguridad marítima, la termografía infrarroja y su aplicación en el ámbito naval, así como el uso de la visión artificial en entornos marítimos, la automatización de

resultados y los términos de *Deep Learning* y redes neuronales aplicados a la detección de objetos. Este capítulo permite contextualizar el trabajo y justificar las decisiones técnicas adoptadas.

2. Desarrollo: tras contextualizarse el caso de estudio, se describen los procedimientos adoptados a lo largo del trabajo. Se detallan las zonas y plataformas de trabajo, los materiales y medios empleados y los objetos de interés considerados para la investigación. Asimismo, se expone la metodología empleada para la adquisición de las imágenes termográficas, la preparación y etiquetado del conjunto de datos y el diseño, aprendizaje y prueba de los modelos de detección automática.
3. Resultados y discusión: en este capítulo se presentan los diferentes resultados obtenidos tras la aplicación de las arquitecturas ‘YOLO’ y ‘Mask R-CNN’ del modelo *Deep Learning* ejecutable propuesto. Finalmente, se realiza una comparación crítica entre ambos modelos, evaluando el comportamiento y rendimiento en el contexto del estudio.
4. Conclusiones: se analiza el grado de cumplimiento de los objetivos planteados y se exponen las limitaciones identificadas. Finalmente, se proponen posibles vías de mejora y líneas futuras de investigación derivadas del trabajo realizado.

2 ESTADO DEL ARTE

2.1 Seguridad marítima a nivel global

La seguridad marítima constituye uno de los pilares fundamentales para el correcto funcionamiento del comercio internacional, la protección de las infraestructuras críticas y la salvaguarda de la vida humana en la mar. Teniendo en cuenta que un alto porcentaje de la población mundial vive dentro de las 100 millas de la costa, que en la actualidad 213 Estados (83% del total mundial) son costeros, que la actividad humana – pesca, comercio, explotación del subsuelo, etc. – se lleva a cabo dentro de una franja de 300 millas contada desde la costa y que más del 80% del transporte mundial de mercancías se realiza por mar (véase Figura 2.1), tiene sentido el concepto de “Seguridad Marítima” como pilar fundamental de la “Seguridad” global, lo que convierte a los océanos y mares en espacios estratégicos de primer orden desde el punto de vista económico, político y de seguridad [1], [2].

En este contexto, la comunidad internacional se enfrenta a un conjunto diverso de riesgos y amenazas que afectan directamente a la seguridad de la navegación. Entre ellos se incluyen los accidentes marítimos, las colisiones entre embarcaciones, la presencia de obstáculos u objetos flotantes, así como actividades ilícitas como el tráfico ilegal, la piratería o la inmigración irregular [3]. A estos factores se suman las condiciones ambientales adversas propias del medio marino, que dificultan la detección temprana de peligros y aumentan la probabilidad de incidentes. Asimismo, las tensiones geopolíticas y los conflictos armados, como evidencian los incidentes recurrentes en el estrecho de Ormuz, han puesto de manifiesto la vulnerabilidad de las rutas marítimas en escenarios de conflicto regional [4].

La creciente densidad del tráfico marítimo, especialmente en zonas costeras, estrechos y áreas portuarias, ha incrementado la complejidad de las operaciones de vigilancia y control. Como consecuencia, la detección eficaz y temprana de objetos de interés en superficie se ha convertido en un elemento clave para garantizar la seguridad marítima, prevenir accidentes y apoyar la toma de decisiones por parte de las autoridades competentes.

Tradicionalmente, la vigilancia marítima se ha apoyado en diferentes medios y sistemas que han demostrado su utilidad, pero a su vez presentan limitaciones significativas en determinados escenarios, como en condiciones de baja visibilidad, durante la noche o en entornos complejos donde la interpretación de la información requiere una elevada carga cognitiva por parte del operador humano.

En este escenario global, existe un interés creciente por el desarrollo e integración de nuevas tecnologías que permitan mejorar la capacidad de detección, identificación y seguimiento de objetos en el entorno marítimo. La combinación de sensores avanzados con técnicas de procesamiento automático de la información y de inteligencia artificial se perfila como una de las líneas de evolución más prometedoras para reforzar la seguridad marítima a nivel mundial [5].

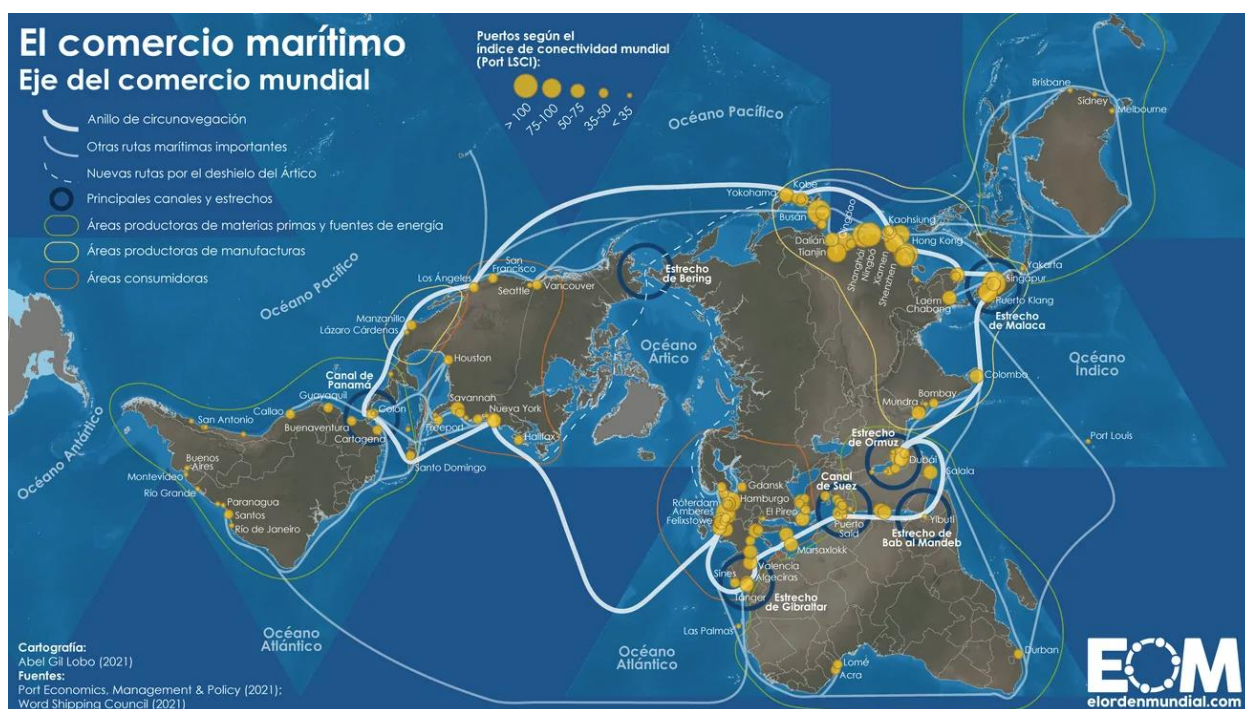


Figura 2.1 El comercio marítimo mundial [6]

2.2 Seguridad y vigilancia marítima en la Armada

España, debido a su situación geoestratégica, su dependencia económica del sector marítimo y a la magnitud de sus espacios marítimos de soberanía e interés, le otorga una importancia fundamental a la seguridad y vigilancia de estos. En este contexto, la Armada Española desempeña un papel esencial en la protección del dominio marítimo, contribuyendo de forma directa a la seguridad nacional y al cumplimiento de los compromisos internacionales adquiridos por el Estado [7].

La vigilancia marítima que aporta la Armada constituye una actividad permanente orientada a la detección, identificación y seguimiento de riesgos y amenazas que puedan afectar a la soberanía, la seguridad de la navegación, las infraestructuras críticas o los intereses nacionales en la mar. Estas tareas se desarrollan tanto en aguas de soberanía y responsabilidad nacional como en escenarios internacionales, en coordinación con otros organismos civiles y militares.

2.2.1 Amenazas y riesgos

El ámbito marítimo constituye un entorno especialmente complejo desde el punto de vista de la seguridad, debido a su gran extensión, a la diversidad de actividades que en él se desarrollan y a la dificultad inherente para mantener una vigilancia continua y exhaustiva. En este contexto, la seguridad marítima se ve condicionada por una amplia variedad de amenazas y riesgos que pueden afectar tanto a la seguridad de la navegación como a los intereses estratégicos del Estado.

Entre los riesgos de carácter no deliberado destacan los accidentes marítimos, como colisiones entre embarcaciones, varadas o incidentes provocados por condiciones meteorológicas adversas. Asimismo, la presencia de objetos flotantes no identificados, embarcaciones de pequeño tamaño o elementos a la deriva supone un peligro significativo para la navegación, especialmente en zonas costeras o de elevado tráfico marítimo. La detección temprana de este tipo de riesgos resulta fundamental para minimizar la probabilidad de accidentes y reducir sus posibles consecuencias. Junto a estos riesgos, el entorno marítimo es también escenario de amenazas de carácter deliberado vinculadas a actividades ilícitas. Entre ellas se incluyen el tráfico ilegal de mercancías y personas, la inmigración irregular, el narcotráfico o el contrabando, actividades que aprovechan la dificultad de control del espacio marítimo y la cercanía a las fronteras marítimas [8]. A estas amenazas se suma la necesidad de proteger infraestructuras críticas

situadas en el ámbito marítimo, como puertos, instalaciones energéticas o rutas de comunicación esenciales.

El análisis detallado de incidentes de seguridad marítima en diferentes regiones geográficas realizado por el Centro de Operaciones y Vigilancia de Acción Marítima (COVAM) [9] de la Armada Española revela que la distribución de incidentes muestra variaciones significativas según el área de interés. En zonas como África Occidental y el Golfo de Guinea, o India, Asia y el Estrecho de Malaca, se registra una mayor frecuencia de incidentes relacionados con delincuencia marítima, intentos de abordaje y otros sucesos que afectan a la seguridad del tráfico marítimo. Estas tendencias contrastan con otras áreas donde la incidencia es menor o predominan riesgos de carácter diferente, como condiciones ambientales adversas o avistamientos sospechosos. La presencia de este patrón geográfico en los incidentes pone de manifiesto que las amenazas no están uniformemente distribuidas, sino que se concentran en corredores de alto tráfico y zonas con elevados niveles de actividad humana o criminal, lo que incrementa la complejidad de la vigilancia marítima y la necesidad de sistemas adaptativos de detección.

En consecuencia, la identificación y gestión de amenazas y riesgos en el entorno marítimo exige el empleo de sistemas de vigilancia capaces de operar de forma fiable en un amplio abanico de escenarios y condiciones ambientales. La progresiva complejidad del entorno de amenazas y la evolución tecnológica refuerzan la necesidad de analizar y desarrollar nuevas soluciones que incrementen la eficacia de los sistemas de vigilancia y detección que complementen los sistemas tradicionales y permitan mejorar la eficacia global de las operaciones de seguridad y vigilancia marítima.

2.2.2 Marco estratégico de seguridad nacional y marítima

La actuación de la Armada Española en materia de seguridad marítima se encuentra enmarcada dentro de un conjunto de documentos estratégicos que definen las prioridades y líneas de actuación del Estado en este ámbito. Entre ellos destacan la Estrategia de Seguridad Nacional [10] y la Estrategia Nacional de Seguridad Marítima [11], que establecen los principios fundamentales para la protección del dominio marítimo y de los intereses asociados al mismo.

Estas estrategias subrayan la necesidad de garantizar la libertad de navegación, proteger las infraestructuras críticas marítimas y reforzar las capacidades de vigilancia y control del espacio marítimo, identificando el ámbito marítimo como uno de los dominios esenciales para la seguridad del Estado. Asimismo, se destaca su especial vulnerabilidad frente a amenazas híbridas, actividades ilícitas y situaciones de riesgo que requieren una vigilancia constante y una capacidad de respuesta eficaz. En este contexto, la detección temprana de objetos y situaciones anómalas en superficie resulta esencial para anticipar riesgos, reducir los tiempos de reacción y apoyar la toma de decisiones operativas por parte de las autoridades competentes.

Los documentos estratégicos también reconocen la importancia de mejorar la conciencia situacional marítima, promoviendo la integración de distintos sensores, sistemas de información y capacidades tecnológicas avanzadas. Dentro de este marco, la Armada Española actúa como uno de los principales instrumentos del Estado para la aplicación de la política de seguridad marítima, tanto en el ámbito nacional como en su participación en operaciones internacionales.

2.2.3 Competencias

Las competencias de la Armada Española en materia de seguridad marítima se ejercen mediante la realización de operaciones tanto en el ámbito nacional como en el internacional, habitualmente en coordinación con otros organismos e instituciones. Estas operaciones abarcan desde tareas de vigilancia, inspección y control, hasta misiones de cooperación internacional y lucha contra amenazas globales. El cumplimiento de estas competencias se realiza tanto de manera autónoma como en coordinación con otros organismos nacionales, como la Guardia Civil, Salvamento Marítimo o las autoridades portuarias, así como con agencias europeas e internacionales. Esta cooperación interinstitucional resulta

fundamental para garantizar una vigilancia eficaz y evitar duplicidades en el empleo de medios y recursos.

En la Figura 2.2 se muestra un esquema resumen de las principales operaciones y ámbitos de actuación de la Armada relacionadas con la seguridad marítima.



Figura 2.2 Competencias y principales operaciones de la Armada en el ámbito de la seguridad marítima (Elaboración propia) [12]

Para el desarrollo de estas misiones, la Armada dispone de una amplia variedad de plataformas y sistemas de vigilancia, que incluyen buques, aeronaves y sensores embarcados y terrestres. No obstante, la complejidad del entorno marítimo y las limitaciones inherentes a los sistemas tradicionales hacen necesario avanzar hacia soluciones que permitan mejorar la detección automática de objetos de interés, reducir la carga de trabajo del operador humano y aumentar la eficacia global de las operaciones de vigilancia marítima.

2.3 Sensores y sistemas empleados en entornos marinos

La vigilancia y el control del entorno marítimo se apoyan en el empleo de distintos sensores y sistemas de detección, contribuyendo a mejorar la conciencia situacional marítima.

2.3.1 Sistemas radar

Los sistemas radar constituyen una de las tecnologías fundamentales en la vigilancia y seguridad marítima, siendo empleados tanto en plataformas navales como en estaciones costeras para detectar, monitorizar y seguir objetos en superficie bajo una amplia gama de condiciones ambientales. Un radar (acrónimo de *Radio Detection and Ranging*) funciona mediante la emisión de pulsos de energía electromagnética y la posterior recepción de las reflexiones producidas por los objetos, permitiendo determinar distancias, rumbos y velocidades relativas de blancos detectados [13]. El esquema de su funcionamiento se puede observar en la Figura 2.3.

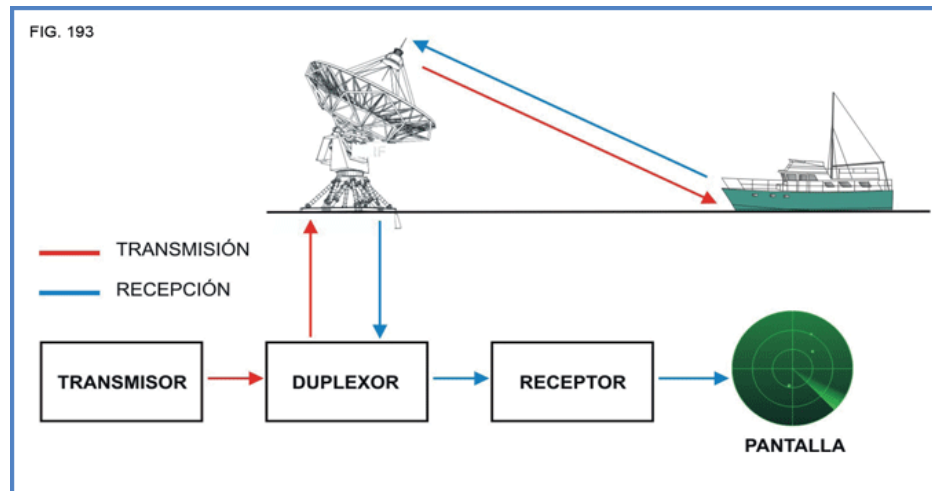


Figura 2.3 Funcionamiento básico de un sistema radar de navegación [14]

Los radares marítimos pueden clasificarse de acuerdo con su función y banda operativa [15]:

- Radar de superficie o *surface-search*, empleado para detectar objetos sobre la superficie del mar, como embarcaciones, boyas o grandes obstáculos, y que es habitual en buques y patrulleras.
- Radares de navegación y vigilancia costera, instalados en tierra o en estructuras portuarias, que proporcionan vigilancia continua de corredores marítimos, entradas a puertos y zonas de interés estratégico.
- Radares de banda X, S y L, según su frecuencia: las bandas X ofrecen alta resolución para objetos pequeños o cercanos, mientras que S y L proporcionan mayor alcance y penetración en condiciones meteorológicas adversas, siendo útiles para vigilancia de áreas extensas o en alta mar.

Los sistemas radar no solo proporcionan información sobre la presencia de objetos en la superficie, sino que también permiten mantener una conciencia situacional continua, integrando sus datos con otros sensores y sistemas de mando y control. Su operación bajo cualquier condición meteorológica les confiere una ventaja significativa respecto a sensores ópticos tradicionales, ya que mantienen funcionalidad en situaciones de niebla, oscuridad o lluvia.

2.3.2 Sistemas de identificación automática de buques (AIS)

El sistema AIS (*Automatic Identification System*) es una herramienta fundamental en la vigilancia y seguridad marítima, diseñada para mejorar la seguridad de la navegación y facilitar el control del tráfico marítimo. Se trata de un sistema cooperativo basado en la transmisión automática de información entre buques y estaciones costeras mediante enlaces de radio en la banda VHF (*Very High Frequency*), permitiendo conocer la identidad, posición, rumbo y velocidad de las embarcaciones equipadas con este sistema (véase su funcionamiento en la Figura 2.4). El AIS fue desarrollado inicialmente como una medida de prevención de colisiones, y su uso es obligatorio, según el Convenio SOLAS de la Organización Marítima Internacional (IMO) [16], para buques mercantes de determinado tonelaje y para buques de pasaje. La información transmitida por el sistema incluye datos estáticos, como el nombre del buque, su número MMSI (*Maritime Mobile Service Identity*) o su tipo, así como datos dinámicos obtenidos a partir de sensores de a bordo, como el sistema de posicionamiento GNSS (*Global Navigation Satellite System*).

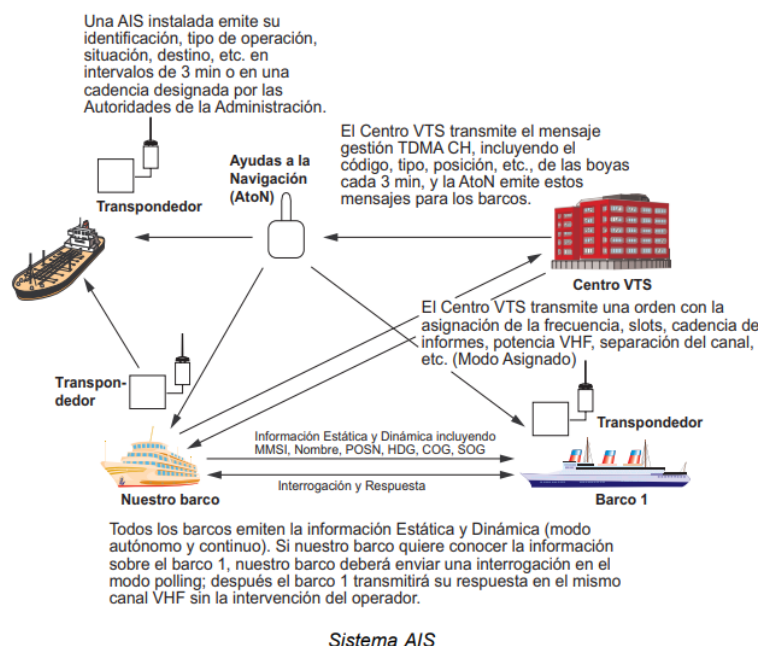


Figura 2.4 Funcionamiento detallado del sistema AIS [17]

2.3.3 Sistemas de seguridad y socorro marítimo (GMDSS)

El Sistema Mundial de Socorro y Seguridad Marítimos (GMDSS, *Global Maritime Distress and Safety System*) es un conjunto integrado de procedimientos, equipos y sistemas de comunicación destinado a garantizar que cualquier buque en situación de emergencia pueda emitir una alerta de socorro y recibir asistencia de forma rápida y eficaz. El GMDSS se apoya en la utilización de diferentes medios de comunicación y transmisión de alertas, tanto terrestres como por satélite, incluyendo radiocomunicaciones en VHF, MF (*Medium Frequency*) y HF (*High Frequency*), sistemas satelitales como Inmarsat, radiobalizas de localización de siniestros (EPIRB – *Emergency Position Indicating Radio Beacon*), respondedores radar (SART – *Search And Rescue Transponder*) y sistemas de llamada selectiva digital (DSC – *Digital Selective Calling*), como puede observarse en la Figura 2.5 [18]. Estos elementos permiten emitir alertas automáticas que incluyen información sobre la posición del buque y la naturaleza de la emergencia, facilitando la coordinación de las operaciones de búsqueda y rescate (*Search and Rescue*, SAR).

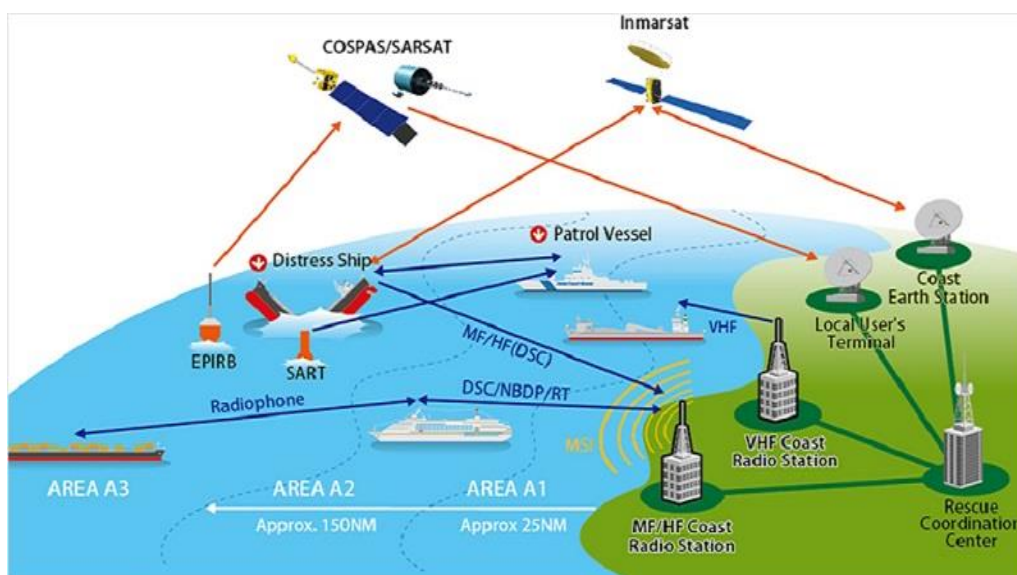


Figura 2.5 Funcionamiento del sistema GMDSS [19]

2.3.4 Sistemas ópticos convencionales

2.3.4.1 Cámaras en el espectro visible

Las cámaras que operan en el espectro visible (véase Figura 2.6) son dispositivos que captan la radiación electromagnética comprendida aproximadamente entre los 400 y 700 nm, correspondiente a la porción del espectro perceptible por el ojo humano, generando imágenes que reproducen de forma directa el entorno observado. Su principal ventaja radica en la facilidad de interpretación de las imágenes obtenidas, ya que presentan una representación visual intuitiva que facilita la identificación y clasificación de objetos por parte de los operadores.

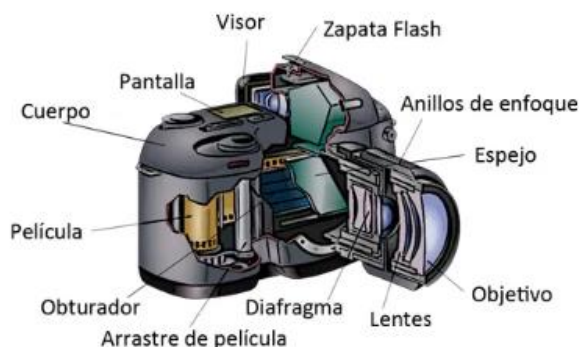


Figura 2.6 Esquema de cámara convencional para espectro visible [20]

En el ámbito marítimo, las cámaras visibles se utilizan ampliamente para tareas de observación, identificación visual de contactos, vigilancia de áreas portuarias y control del tráfico marítimo [21]. Integradas habitualmente en sistemas electroópticos estabilizados, permiten realizar observaciones a larga distancia mediante el uso de ópticas con elevados aumentos y sistemas de estabilización que compensan el movimiento de la plataforma. Estas capacidades las convierten en una herramienta de apoyo fundamental a otros sensores como el radar o el AIS, especialmente en fases finales de identificación de objetivos.

2.3.4.2 Cámaras en el espectro infrarrojo

Las cámaras que operan en el espectro infrarrojo son dispositivos de adquisición de imagen diseñados para captar radiación electromagnética en longitudes de onda superiores al espectro visible, generalmente comprendidas entre los 700 nm y los 14000 nm. A diferencia de las cámaras convencionales, que registran la luz reflejada por los objetos, las cámaras infrarrojas permiten obtener información asociada a la radiación térmica emitida por los cuerpos, invisible al ojo humano, en función de su temperatura, como puede observarse en la Figura 2.7. El principio de funcionamiento de las cámaras infrarrojas se basa en el uso de detectores sensibles a la radiación infrarroja, que convierten la energía electromagnética incidente en señales eléctricas. Estas señales son posteriormente procesadas y transformadas en imágenes digitales, en las que cada píxel representa un nivel de radiación o temperatura relativa [22].

Desde el punto de vista espectral, las cámaras infrarrojas pueden clasificarse en infrarrojo cercano (NIR – *Near Infrared Spectroscopy*), infrarrojo medio (MWIR – *Mid Wave Infrared*) e infrarrojo lejano (LWIR – *Long Wave Infrared*). Las cámaras NIR operan en bandas próximas al visible y suelen emplearse en aplicaciones donde se requiere una transición suave entre ambos espectros. Por su parte, las cámaras MWIR y LWIR, comúnmente denominadas cámaras térmicas, detectan la radiación térmica emitida por los objetos generando imágenes que representan diferencias térmicas relativas en la escena observada.



Figura 2.7 Imagen de un cuadro eléctrico en el espectro infrarrojo tomada con la cámara termográfica modelo IR-282 [23]

La información proporcionada por las cámaras infrarrojas no está ligada a la percepción visual humana, sino que ofrece una representación alternativa del entorno basada en patrones térmicos, lo que permite identificar diferencias que no resultan apreciables en el espectro visible. Esta característica convierte a las cámaras infrarrojas en una herramienta especialmente adecuada para tareas de observación y análisis en condiciones de baja iluminación o ausencia total de luz, así como para el estudio de fenómenos donde la temperatura constituye una variable relevante. En el ámbito marítimo, las cámaras térmicas MWIR y LWIR son ampliamente utilizadas para la detección de embarcaciones, personas u objetos flotantes sobre la superficie del mar, aprovechando el contraste térmico existente entre dichos objetivos y el entorno marino. Estas cámaras resultan especialmente eficaces en tareas de vigilancia nocturna, protección de infraestructuras portuarias, control de accesos marítimos y apoyo a operaciones de búsqueda y rescate. La capacidad de representar la información térmica facilita la detección de elementos que podrían pasar inadvertidos en el espectro visible, especialmente en situaciones de baja visibilidad.

2.3.5 Limitaciones

Los sistemas de detección y vigilancia empleados actualmente en el ámbito marítimo presentan una serie de limitaciones inherentes a su naturaleza y a las condiciones del entorno en el que operan. Aunque tecnologías como el radar, el AIS, el GMDSS y los sensores ópticos proporcionan información valiosa y complementaria, en ocasiones no resultan suficientes por sí solas para garantizar una vigilancia continua, fiable y completa en todos los escenarios operativos.

Los sistemas radar, pese a su capacidad de detección todo tiempo, pueden presentar dificultades en la identificación y clasificación precisa de objetivos, especialmente en presencia de eco marino, interferencias o blancos de pequeño tamaño. Por su parte, el AIS depende principalmente de que los buques transmitan voluntariamente su información, por lo que puede ser desconectado, manipulado o falsificado de forma intencionada, dificultando su correcta identificación y clasificación [24]. Asimismo, embarcaciones de pequeño tamaño u objetos flotantes que no disponen de AIS, quedan fuera de la capacidad de detección. Estas limitaciones hacen necesario complementar el AIS con otros sensores, como el radar o los sistemas ópticos. El GMDSS, aunque esencial para la gestión de emergencias, no constituye un sistema de detección preventiva, ya que su activación se produce una vez declarada la situación de socorro.

En cuanto a los sistemas ópticos, tanto en el espectro visible como en el infrarrojo, su eficacia se ve condicionada por la necesidad de una supervisión humana continua para la interpretación de las imágenes. La dependencia del operador introduce una carga cognitiva elevada y limita la capacidad de

análisis simultáneo de grandes volúmenes de información visual, especialmente en escenarios prolongados o de elevada complejidad.

A pesar de estas limitaciones, las cámaras en el espectro infrarrojo representan una mejora significativa frente a los sistemas ópticos visibles en términos de continuidad operativa y robustez ante condiciones desfavorables. Su capacidad para proporcionar información térmica del entorno, sin verse afectada por la presencia de eco marino, interferencias o blancos de pequeño tamaño como en el caso de los sistemas radar, las convierte en un elemento clave dentro de los sistemas modernos de vigilancia marítima y sienta las bases para la aplicación de técnicas avanzadas de procesamiento de imágenes y detección automática de objetos, aspecto desarrollado en apartados posteriores de este trabajo.

2.4 Termografía infrarroja

2.4.1 Generalidades

La termografía infrarroja, también llamada de forma breve termografía, es una técnica de adquisición y análisis de imágenes basada en la detección de la radiación infrarroja emitida de forma natural por los cuerpos en función de su temperatura. A diferencia de las técnicas ópticas convencionales, la termografía no depende de la iluminación externa ni de la radiación reflejada. De acuerdo con los principios de la termodinámica, todo cuerpo posee una temperatura superior al cero absoluto (0 K, $-273,15\text{ }^{\circ}\text{C}$) y, como consecuencia, emite radiación electromagnética. La cantidad de energía radiada y su distribución espectral dependen principalmente de la temperatura del objeto y de sus propiedades superficiales, en particular de la emisividad, un concepto fundamental en la interpretación de las imágenes termográficas, definida como la capacidad que tiene un objeto de emitir radiación térmica en comparación con la de un cuerpo negro perfecto a la misma temperatura y longitud de onda. La emisividad es un parámetro adimensional comprendido entre 0 y 1, que depende de las propiedades físicas y superficiales del material, así como de la longitud de onda considerada. Las cámaras termográficas están diseñadas para captar esta radiación en determinadas bandas del espectro infrarrojo y convertirla en una señal eléctrica que posteriormente se transforma en una imagen digital interpretable [25].

Las imágenes termográficas representan la intensidad de la radiación infrarroja detectada mediante una escala de colores o niveles de gris, en la que cada píxel corresponde a un valor térmico relativo o a una temperatura aparente, como puede verse en la Figura 2.8. Esta representación permite cuantificar (con temperaturas absolutas), identificar patrones térmicos, gradientes de temperatura y diferencias térmicas entre distintos elementos de una escena, incluso cuando dichos elementos no resultan distinguibles mediante técnicas de imagen convencionales.

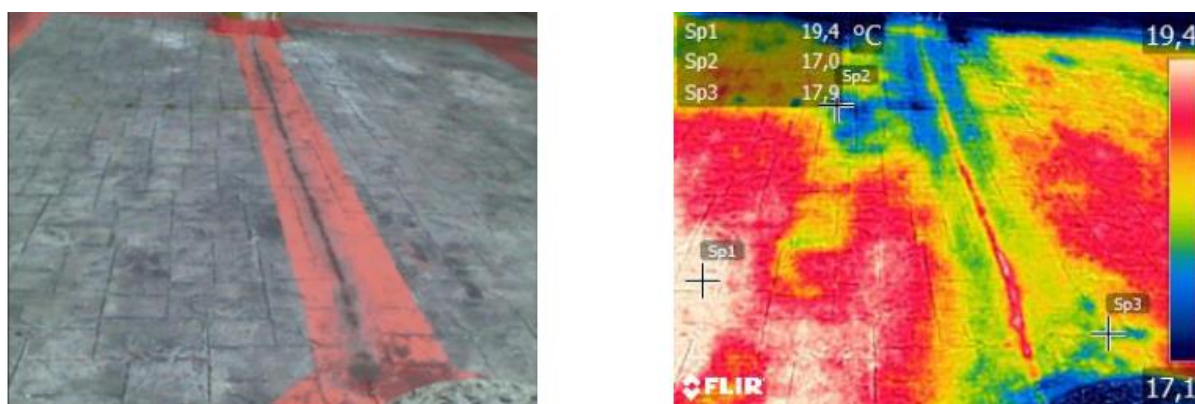


Figura 2.8 Ejemplo de imagen termográfica [25]

Una de las características fundamentales de la termografía infrarroja es su carácter pasivo, ya que no requiere la emisión de energía hacia el objeto observado. Esta propiedad permite su utilización en una amplia variedad de condiciones ambientales y operativas, independientemente de la presencia o

ausencia de iluminación visible. Asimismo, la termografía posibilita la observación remota y sin contacto, lo que amplía su campo de aplicación en numerosos ámbitos científicos y tecnológicos.

Como se ha comentado previamente, la termografía infrarroja no proporciona, en general, una medición directa de la temperatura real de los objetos, sino una estimación influida por factores como la emisividad de los materiales, la transmisión atmosférica o las condiciones de adquisición. No obstante, para muchas aplicaciones el interés principal reside en el análisis cualitativo o relativo de los contrastes térmicos, más que en la obtención de valores absolutos de temperatura. Estas características convierten a la termografía infrarroja en una herramienta versátil para el estudio y análisis de fenómenos térmicos.

2.4.2 Principios físicos

Las leyes de la física en las que se basa la termografía infrarroja se fundamentan en las de radiación electromagnética, las cuales permiten establecer una relación cuantitativa entre la energía radiada y la temperatura del objeto. Esta radiación se genera por la aceleración y desaceleración de partículas atómicas cargadas. Es una técnica de ensayo no destructivo y no invasiva que visualiza y mide la radiación térmica emitida por superficies, permitiendo mapear temperaturas sin contacto físico.

El análisis teórico se fundamenta en varias leyes físicas y conceptos clave [26]:

- Longitud de onda $\rightarrow$ la energía electromagnética se transporta como una onda con velocidad (c), longitud de onda (λ) en metros y frecuencia (f) en Hercios, siendo c la velocidad de la luz de 3×10^8 metros/segundo (Ecuación 1):

$$\lambda = \frac{c}{f} \quad \text{Ecuación 1}$$

La longitud de onda de la zona infrarroja del espectro electromagnético se encuentra entre 0,7 y 14 micras (véase Figura 2.9).

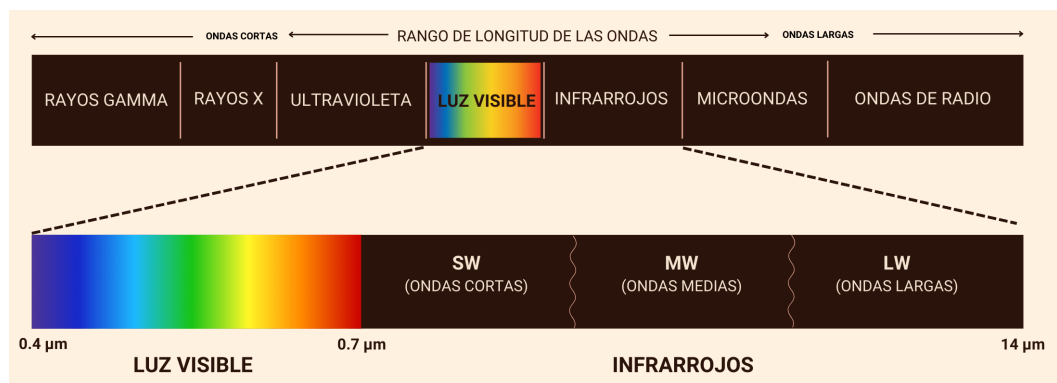


Figura 2.9 Espectro electromagnético [27]

Las leyes de radiación infrarroja se basan en el comportamiento de un cuerpo negro, es decir, un objeto teórico que emite la máxima cantidad de energía en todas las temperaturas y longitudes de onda. Por lo tanto, es un emisor y absorbente perfecto de energía.

- Ley de Stephan-Boltzmann $\rightarrow$ explica la sensibilidad a variaciones térmicas, determinando que la cantidad de energía radiada por un objeto es proporcional a la temperatura del objeto (en grados *Kelvin*) elevada a la cuarta potencia (Ecuación 2), medida en $W/m^2 \cdot K^4$:

$$\sigma = 5,67 \times 10^{-8} \times T^4 \quad \text{Ecuación 2}$$

- Ley de Planck → describe la distribución espectral de la radiación emitida por un cuerpo negro ideal (véase Figura 2.10). Planck postuló que la energía no se emite de forma continua, sino en paquetes llamados cuantos. Por ello la fórmula de la energía (E , medida en Julios) (Ecuación 3), siendo h la constante de Planck ($6,626 \times 10^{-34} \text{ J} \cdot \text{s}$) y ν la frecuencia de la radiación, medida en Hercios:

$$E = h \times \nu$$

Ecuación 3

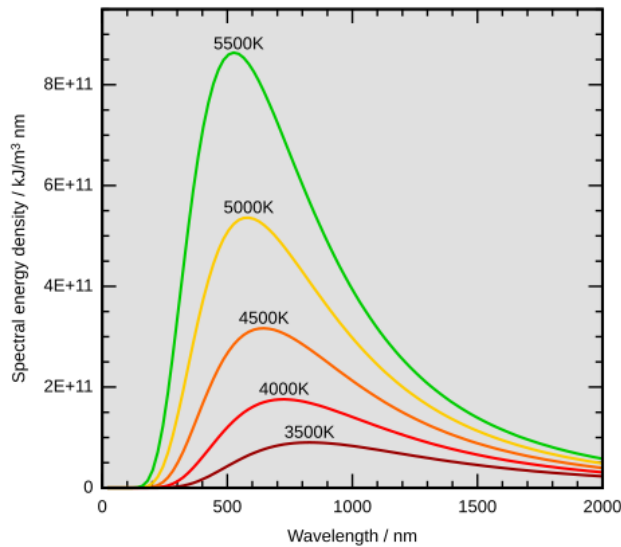


Figura 2.10 Ley de Planck para cuerpos a diferentes temperaturas [28]

- Ley de Wien → justifica la relación entre la temperatura de un cuerpo y la longitud de onda a la que se produce la máxima emisión de radiación térmica. A medida que aumenta la temperatura, dicha longitud de onda se desplaza hacia valores menores, lo que explica que los objetos a temperatura ambiente emitan principalmente en el rango LWIR, mientras que a temperaturas más elevadas la emisión máxima se aproxima al rango MWIR (Ecuación 4):

$$\lambda^{max} = \frac{2898}{T}$$

Ecuación 4

2.4.3 Modos de operación

Según exista o no estimulación externa de calor hacia el objeto inspeccionado, la termografía infrarroja opera principalmente bajo dos modos fundamentales: pasivo y activo, cuyas características se muestran en la Tabla 2-1. Del mismo modo, existen dos tipos de termografía según el enfoque del análisis que se desee realizar: cualitativa y cuantitativa [29].

- El modo pasivo se basa en la medición de la radiación infrarroja natural que emite un objeto debido a su propio funcionamiento, proceso o condiciones ambientales, mientras que el modo activo consiste en aplicar una fuente de calor externa o estímulo al objeto para crear una diferencia de temperatura (contraste térmico) entre la zona sana y los defectos internos.
- En la termografía cualitativa se hace un enfoque para identificar patrones térmicos y anomalías visuales en la imagen, como "puntos calientes" o "zonas frías", siendo un análisis rápido para detectar anomalías relativas. La termografía cuantitativa mide y analiza valores de temperatura

precisos para determinar la severidad de un problema, comparando los datos con referencias o líneas base.

Característica	Pasiva	Activa
Fuente de calor	Natural / Propia del objeto	Externa (óptica, mecánica, electromagnética y eléctrica)
Objetivo	Inspección en servicio, mantenimiento	Detección de defectos internos
Complejidad	Baja-Media	Alta (requiere estimulación)
Aplicaciones	Paneles, edificios, motores	Compuestos, aeronáutica, metal

Tabla 2-1 Resumen de diferencias entre la termografía activa y pasiva (Elaboración propia)

2.4.4 Aplicación en el ámbito marítimo

En la actualidad, la termografía infrarroja se emplea de forma extendida en actividades de investigación y desarrollo, así como en numerosos sectores industriales, tales como ensayos no destructivos, monitorización de condiciones y mantenimiento predictivo. La evolución tecnológica de los sensores infrarrojos y, especialmente, la aparición de dispositivos de menor coste ha favorecido la expansión de esta técnica hacia nuevos campos de aplicación, incrementando su accesibilidad y versatilidad.

En el ámbito marítimo, las cámaras térmicas MWIR y LWIR previamente mencionadas son habitualmente utilizadas para la detección de embarcaciones, personas u objetos flotantes sobre la superficie del mar, aprovechando el contraste térmico existente entre dichos objetivos y el entorno marino, lo que permite su identificación incluso en condiciones de iluminación reducida o durante la noche, como puede observarse en la Figura 2.11. Como consecuencia, estas cámaras resultan especialmente eficaces en tareas de vigilancia marítima, protección de infraestructuras portuarias, control de accesos y apoyo a operaciones de búsqueda y rescate. Los sistemas infrarrojos empleados en aplicaciones navales suelen integrarse en plataformas electroópticas estabilizadas, que combinan sensores visibles e infrarrojos en una misma unidad. Esta configuración permite al operador alternar entre distintos modos de observación y aprovechar las ventajas de cada espectro según el escenario operativo. Asimismo, los avances en tecnología de detectores, óptica infrarroja y procesamiento digital han permitido mejorar de forma significativa la resolución espacial, la sensibilidad térmica y la estabilidad de estos sistemas, ampliando su eficacia en entornos marítimos complejos [30].



Figura 2.11 Comparativa de imágenes del espectro visible e infrarrojo de una cámara FLIR en la mar [30]

Diversos estudios recientes destacan además el creciente interés por la aplicación de la termografía infrarroja en sistemas marítimos, tanto para tareas de vigilancia como para la monitorización del estado

de sistemas y el mantenimiento predictivo de equipos embarcados. En particular, revisiones sistemáticas de la literatura han señalado que la termografía permite detectar de forma temprana anomalías térmicas asociadas a fallos en componentes eléctricos o mecánicos, contribuyendo a mejorar la fiabilidad operativa y la seguridad de los sistemas marítimos [31]. Asimismo, se han desarrollado sistemas de teledetección multispectral específicamente diseñados para aplicaciones marítimas. Un ejemplo es el sistema MACS-Mar (*Modular Airborne Camera System*), que combina sensores en el espectro visible, infrarrojo cercano, hiperspectral y térmico con algoritmos de detección automática para identificar embarcaciones, objetos flotantes o anomalías en la superficie del mar en tiempo real. Este tipo de plataformas permite generar información georreferenciada y transmitirla a centros de control para mejorar la conciencia situacional marítima y apoyar la toma de decisiones operativas [32].

La creciente incorporación de cámaras infrarrojas en sistemas de vigilancia marítima ha favorecido también el desarrollo de técnicas avanzadas de análisis de imagen orientadas a la detección y clasificación automática de objetos. En este contexto, la información térmica proporcionada por estos sensores constituye una base especialmente adecuada para la aplicación de algoritmos de procesamiento de imágenes y modelos de inteligencia artificial, aspecto que adquiere una relevancia creciente en los sistemas modernos de seguridad y vigilancia marítima.

2.5 Inteligencia Artificial en la detección de objetos

2.5.1 Introducción a la IA

La Inteligencia Artificial (IA) es la capacidad de las máquinas para aprender a partir de los datos y utilizar lo aprendido en la toma de decisiones de forma similar a como lo haría un ser humano. En otras palabras, es la habilidad de los ordenadores para realizar tareas que normalmente requieren inteligencia humana, con la excepción de que los dispositivos basados en IA no necesitan descansar, pudiendo analizar grandes volúmenes de datos al mismo tiempo [33].

Debido a sus capacidades, la IA está siendo utilizada en numerosas situaciones sirviendo como ayuda a los humanos para beneficiarse de mejoras significativas y disfrutar de una mayor eficiencia en casi todos los ámbitos de la vida. Asimismo, dos de las aplicaciones técnicas de la IA que están creciendo rápidamente en la actualidad son el reconocimiento de imágenes, clasificación y etiquetado (véase Figura 2.12), y la detección y clasificación de objetos, herramientas útiles para una variada gama de industrias, así como para el objeto del presente trabajo.



Figura 2.12 Categorías y aplicaciones actuales más comunes de la IA [34]

En el contexto del procesamiento de imágenes, la IA permite automatizar el análisis de información visual mediante algoritmos capaces de extraer características relevantes y establecer relaciones complejas entre los datos de entrada, véase la Figura 2.13. A diferencia de los enfoques clásicos, basados en reglas fijas o umbrales definidos manualmente, los sistemas basados en IA pueden adaptarse al entorno y mejorar su rendimiento a medida que se entrenan con nuevos conjuntos de datos. Esta capacidad de aprendizaje resulta especialmente valiosa en escenarios complejos y variables, donde la definición explícita de reglas resulta limitada o ineficaz.

En aplicaciones de vigilancia y detección, la IA permite reducir la carga cognitiva del operador humano al filtrar grandes volúmenes de información y destacar únicamente aquellos elementos relevantes. De este modo, los sistemas inteligentes actúan como una herramienta de apoyo a la toma de decisiones, mejorando la eficiencia y la fiabilidad de los sistemas de detección y vigilancia. Esta capacidad resulta clave para la evolución hacia soluciones más automatizadas y robustas.

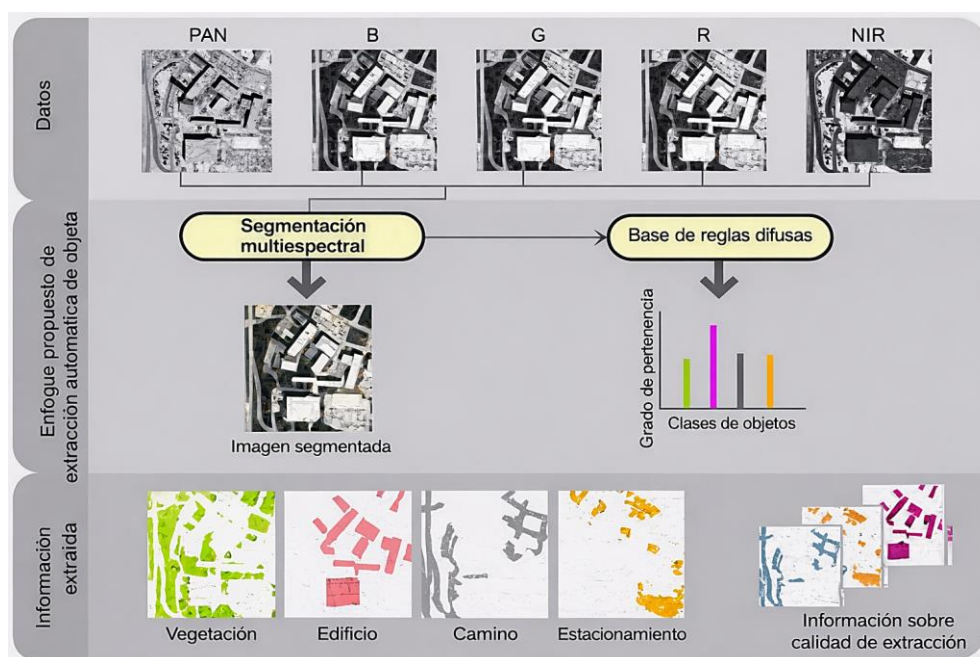


Figura 2.13 Procesamiento de imágenes para la extracción de objetos [35]

2.5.2 Aprendizaje automático (Machine Learning)

El *Machine Learning* (ML) es un campo de la Inteligencia Artificial centrado en el desarrollo de algoritmos capaces de aprender patrones y relaciones a partir de datos, mejorando su rendimiento de forma progresiva sin necesidad de ser programados explícitamente para cada tarea. A diferencia de los enfoques tradicionales, en los que el comportamiento del sistema se define mediante reglas fijas, el aprendizaje automático permite que el modelo se adapte a partir de ejemplos representativos del problema a resolver. En el ámbito del procesamiento de imágenes, el aprendizaje automático se utiliza para abordar tareas como la clasificación, la detección y la segmentación de objetos. Estos algoritmos operan a partir de conjuntos de datos etiquetados o no etiquetados, extrayendo características relevantes de las imágenes y utilizando dichas características para establecer modelos predictivos. El proceso de entrenamiento consiste en ajustar los parámetros del modelo para minimizar el error entre las predicciones realizadas y los valores reales proporcionados durante la fase de aprendizaje [36].

Los métodos de aprendizaje automático pueden clasificarse en los siguientes [37]:

- Aprendizaje supervisado: a partir de datos etiquetados.
- Aprendizaje no supervisado: a partir de datos no etiquetados.
- Aprendizaje semisupervisado: aprendizaje a partir de datos etiquetados y datos no etiquetados.

- Aprendizaje por refuerzo: a partir de las interacciones con su entorno mediante prueba y error y un sistema de recompensas.

El ML utiliza técnicas matemáticas y estadísticas para analizar y procesar datos, consiguiendo un gran potencial para revolucionar la forma en que se interactúa con el mundo. Sin embargo, aunque los métodos clásicos de aprendizaje automático han demostrado buenos resultados en determinadas aplicaciones de visión artificial, su rendimiento suele depender en gran medida de la calidad de las características extraídas manualmente antes del aprendizaje de los distintos modelos de ML. Esta limitación ha impulsado el desarrollo de enfoques más avanzados, capaces de aprender de forma automática representaciones jerárquicas directamente a partir de los datos de entrada en bruto, lo que da lugar al aprendizaje profundo (*Deep Learning*), tratado en el apartado siguiente.

2.5.3 Aprendizaje profundo (*Deep Learning*)

El *Deep Learning* (DL) es un tipo especializado de *Machine Learning*, consiste en un modelo de aprendizaje por capas que procesan la información dada en múltiples etapas sucesivas, para así tener en cuenta interacciones complejas entre los datos observados. Este método de aprendizaje se basa en redes neuronales artificiales con múltiples capas ocultas, como se puede observar en la Figura 2.14, donde las redes son sistemas que imitan el funcionamiento del mecanismo del cerebro, intentando aproximarse a nuestra forma de aprender. La ventaja de los modelos DL en comparación con los modelos ML tradicionales proviene del proceso de extracción de características, que se desarrolla de manera autónoma en los modelos DL, a diferencia de los modelos ML tradicionales, donde es más complejo y requiere un conocimiento previo más detallado del dominio del problema antes de la extracción [38].

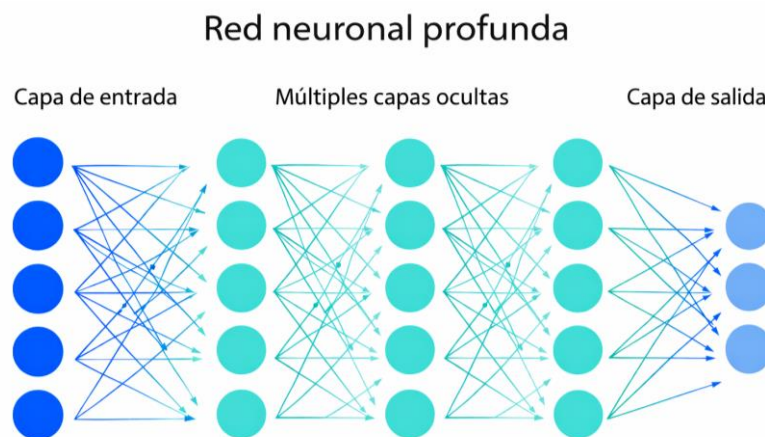


Figura 2.14 Algoritmo de *Deep Learning* basado en una red neuronal [38]

El aprendizaje profundo ha supuesto un avance significativo en el contexto del procesamiento de imágenes, ya que permite abordar problemas complejos como la detección, clasificación y segmentación de objetos con un alto grado de precisión. Las capas sucesivas de una red profunda aprenden representaciones cada vez más abstractas de la información visual, comenzando por patrones simples, como bordes o texturas, y avanzando hacia estructuras más complejas y semánticamente significativas. Una de las principales ventajas del aprendizaje profundo es su capacidad para generalizar en entornos complejos y variables, donde los métodos tradicionales presentan limitaciones. Esta característica resulta especialmente relevante en aplicaciones de visión artificial basadas en imágenes reales, donde las condiciones de adquisición, iluminación o ruido pueden variar de forma significativa.

Como consecuencia, el aprendizaje profundo se ha consolidado como el enfoque predominante en los sistemas modernos de detección automática de objetos, constituyendo la base de arquitecturas especializadas como las redes neuronales convolucionales.

2.6 Redes Neuronales Convolucionales

Las redes neuronales artificiales, previamente mencionadas, son modelos computacionales de inteligencia artificial inspirados en la estructura del cerebro humano, diseñados para reconocer patrones complejos, clasificar datos y aprender de ellos. Compuestas por nodos interconectados (neuronas) organizados en capas (entrada, ocultas y salida), estas redes procesan información ajustando pesos sinápticos durante el aprendizaje [39]. Mediante procesos de entrenamiento basados en la minimización del error, las redes neuronales ajustan sus parámetros y adquieren la capacidad de resolver problemas de clasificación, regresión y reconocimiento de patrones. Sobre esta base se desarrolla la arquitectura de las redes neuronales convolucionales, orientada específicamente al tratamiento de datos con estructura espacial, como las imágenes.

2.6.1 Fundamentos

Las redes neuronales convolucionales (CNN), por sus siglas en inglés *Convolutional Neural Network*; son un tipo de red neuronal artificial con aprendizaje supervisado, diseñados para capturar y aprender patrones complejos presentes en imágenes digitales. Estas redes explotan la organización bidimensional de las imágenes, lo que permite extraer características relevantes de forma jerárquica y eficiente, preservando la información espacial de la escena. Además, las CNN realizan tres tareas principales en el ámbito de la visión artificial: detección y categorización de objetos, clasificación de escenas y clasificación de imágenes en general.

El elemento fundamental de una CNN es la operación de convolución, mediante la cual se aplican filtros o núcleos (*kernels*) sobre la imagen de entrada. Estos filtros recorren la imagen realizando operaciones locales que permiten detectar patrones básicos, como bordes, contornos o texturas, generando mapas de características (*feature maps*). Los parámetros de estos filtros se ajustan automáticamente durante el proceso de entrenamiento. Su arquitectura, la cual se muestra en la Figura 2.15, está compuesta por una secuencia de capas, donde cada capa actúa como un bloque de procesamiento definido por tres elementos fundamentales: entrada, pesos y salida, donde la salida se convierte en la entrada de la siguiente. Estas capas se entienden como un conjunto de filtros comúnmente llamados campos receptivos, los cuales se ajustan para la extracción de características de una señal. A diferencia de una red neuronal tradicional, en la que cada neurona de una capa se conecta a todas las neuronas de la otra, conocido como *Fully Connected* (FC), las CNN emplean mecanismos de compartición de pesos. Esta característica reduce de forma significativa el número de parámetros del modelo y permite extraer información relevante de las imágenes de entrada de manera más eficiente [40].

En las capas más profundas de la red, las características extraídas evolucionan desde patrones simples hacia representaciones progresivamente más abstractas y complejas, lo que permite a la CNN identificar objetos o regiones de interés dentro de la imagen. Finalmente, las capas finales de la arquitectura se encargan de realizar las tareas de clasificación o detección a partir de la información previamente extraída.

En conjunto, las redes neuronales convolucionales constituyen una herramienta especialmente adecuada para el análisis automático de imágenes, al combinar la capacidad de aprendizaje de las redes neuronales con un tratamiento eficiente de la información espacial. Su arquitectura jerárquica y el uso de operaciones de convolución permiten extraer características relevantes de forma progresiva, reduciendo la necesidad de diseño manual de descriptores y aumentando la robustez frente a variaciones en la escena. Estas propiedades han consolidado a las CNN como el enfoque predominante en los sistemas modernos de visión artificial, sirviendo de base para el desarrollo de modelos de detección automática de objetos como los empleados en el presente trabajo.

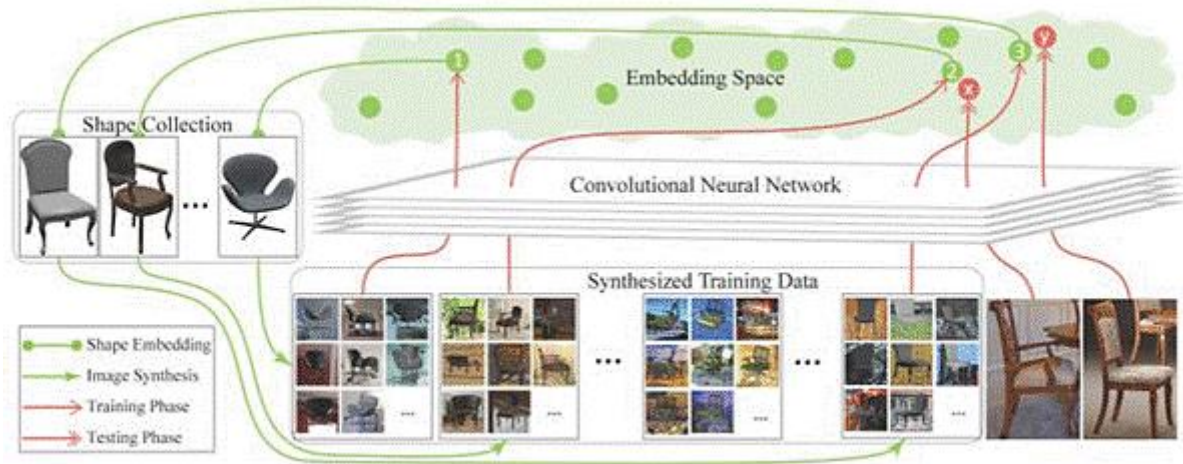


Figura 2.15 Esquema de funcionamiento de la CNN [41]

2.6.2 Aplicación en imágenes y detección de objetos

A pesar de la reciente aparición de los sistemas basados en redes neuronales convolucionales, éstos se encuentran en continuo desarrollo y se han consolidado como una de las herramientas más eficaces para el análisis automático de imágenes y la detección de objetos. Esto se debe principalmente a sus capacidades para analizar grandes volúmenes de información visual y la posterior extracción de patrones relevantes. La información proporcionada tras la aplicación de estos sistemas es clave para la toma de decisiones y la mejora de la conciencia situacional, debido a la reducción de tiempos y la optimización del trabajo a realizar.

Su funcionamiento técnico se basa en el procesamiento de imágenes como matrices, aplicando filtros convolucionales para reconocer bordes, formas y texturas, lo que permite diferenciar entre distintos tipos de objetos. La clasificación de imágenes etiqueta la misma, mientras que la detección identifica la posición exacta (x, y, ancho, alto) de múltiples objetos específicos dentro de una escena [42].

Entre los modelos más populares se encuentran los siguientes:

- YOLO (*You Only Look Once*) → reconocido por su alta velocidad, procesa la imagen en una sola pasada, siendo ideal para aplicaciones en tiempo real (véase Figura 2.16).

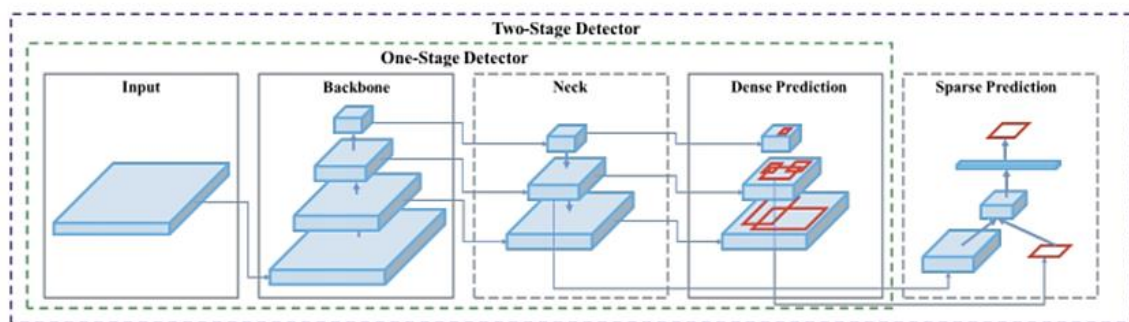


Figura 2.16 Arquitectura de red del modelo YOLO [43]

- MASK R-CNN → basado en la arquitectura *R-CNN* (*Region-based Convolutional Neural Networks*), es un modelo de detección y segmentación de instancias. Primero detecta objetos mediante las cajas delimitadoras (*bounding boxes*) y los clasifica por clases, de forma similar al modelo YOLO, con la diferencia de que, posteriormente, genera una máscara binaria píxel a píxel (*mask*) para cada objeto detectado. Destaca por su alta precisión en la delimitación de contornos, aunque con mayor coste computacional que los detectores de una sola etapa. Se puede observar su esquema de funcionamiento en la Figura 2.17.

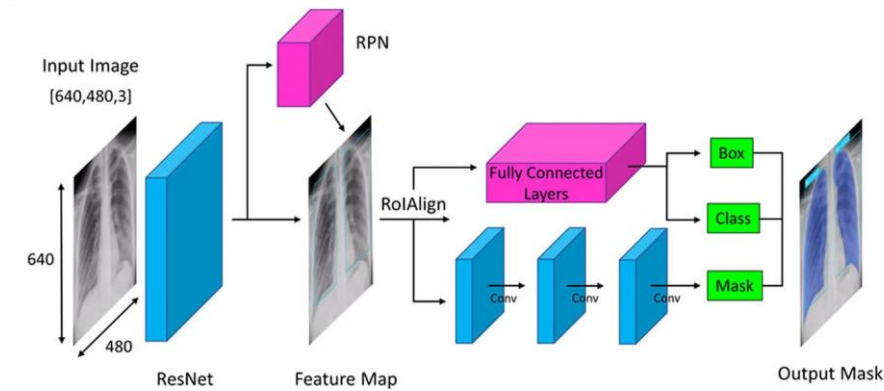


Figura 2.17 Vista general de la arquitectura Mask R-CNN [44]

- SSD (*Single Shot Detector*) → similar a YOLO, ya que predice directamente clases y *bounding boxes* en una única pasada (véase Figura 2.18). Extrae características y varias capas adicionales que detectan objetos a diferentes escalas. Destaca por su equilibrio entre velocidad y precisión, siendo adecuado para aplicaciones en tiempo real.

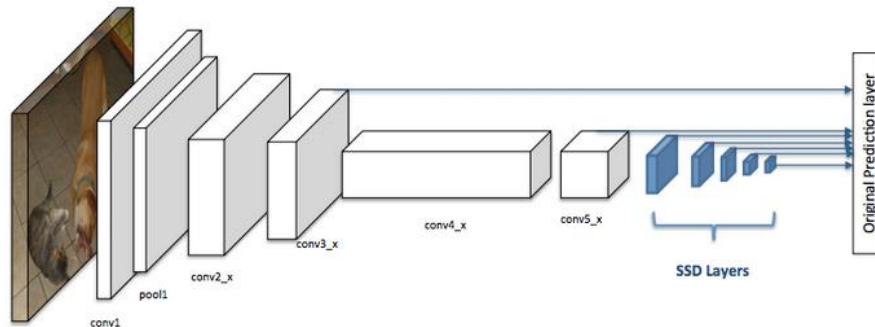


Figura 2.18 Arquitectura de red neuronal convolucional con SSD [45]

- U-NET → diseñada específicamente para segmentación semántica, caracterizada por su arquitectura en forma de “U”, como se observa en la Figura 2.19. Similar a Mask R-CNN, ya que no detecta objetos mediante *bounding boxes*, sino que realiza clasificación píxel a píxel. Es precisamente eficaz cuando se requiere una delimitación precisa de contornos.

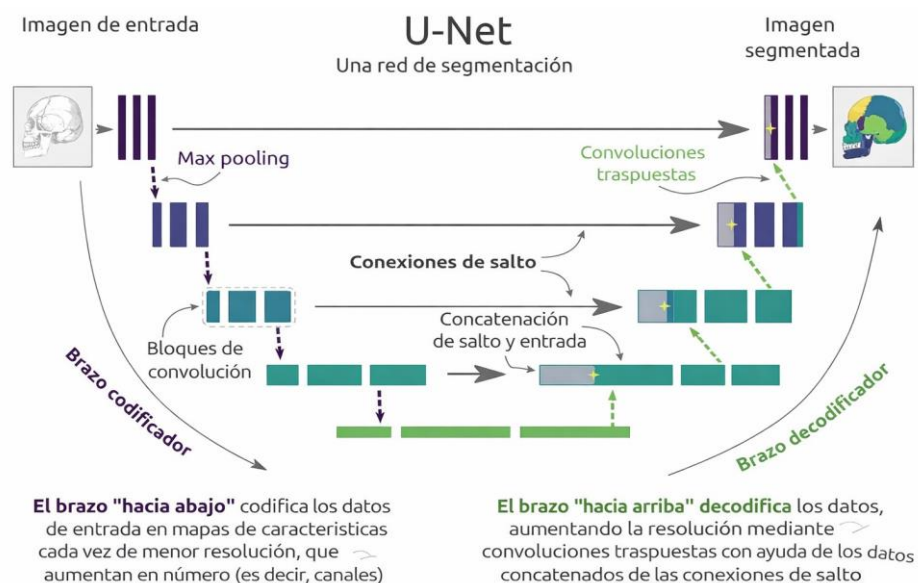


Figura 2.19 Arquitectura de red neuronal U-Net [46]

El modelo YOLO reformula el problema de la detección como una única tarea de regresión directa desde la imagen completa hacia las cajas delimitadoras (*bounding boxes*) y sus probabilidades de clase. Este enfoque convierte la detección en un problema de regresión global optimizado mediante una función de pérdida compuesta que penaliza errores en la localización (coordenadas), tamaño de la caja, confianza y clasificación. A diferencia de los métodos tradicionales basados en propuestas de regiones (como Mask R-CNN), YOLO realiza la detección en una sola pasada por la red, lo que permite un procesamiento significativamente más rápido [47].

Por otra parte, el modelo Mask R-CNN (segundo modelo escogido para su aplicación en imágenes), pertenece a la familia de detectores de dos etapas (*two-stage detectors*), en los que primero se generan propuestas de regiones candidatas y posteriormente se clasifican mediante una red neuronal convolucional [48].

En términos generales, YOLO prioriza la eficiencia computacional y la detección en tiempo real, mientras que los modelos basados en Mask R-CNN priorizan la precisión y la localización detallada de los objetos, como puede observarse en la Tabla 2-2. La elección entre ambos enfoques depende del equilibrio requerido entre velocidad y exactitud, así como de los recursos computacionales disponibles y la naturaleza de la aplicación.

Característica	Mask R-CNN	YOLO
Tipo	Dos etapas	Una etapa
Velocidad	Media	Alta
Precisión	Muy alta	Alta
Detección de objetos pequeños	Mejor	Normalmente inferior
Complejidad	Elevada	Menor

Tabla 2-2 Comparación de características de los modelos Mask R-CNN y YOLO (Elaboración propia)

El uso de redes neuronales convolucionales en la detección de objetos en base a imágenes ha experimentado un notable desarrollo en los últimos años, impulsado por la disponibilidad de grandes conjuntos de datos etiquetados y por el avance de las arquitecturas específicas necesarias. Sin embargo, esta área se ha focalizado mayormente en las imágenes del espectro visible, ya que la mayoría de las cámaras y aplicaciones funcionan en dicho espectro. Este aspecto puede ser contraproducente, ya que al limitar el espectro de imagen a un rango tan pequeño como el visible puede derivar en dejar información relevante fuera de la imagen, sin olvidar que estas técnicas dependen del control de la iluminación. Tareas como son las de búsqueda y rescate (SAR, *Search And Rescue*), vigilancia o seguridad no pueden permitirse el precio de no detectar individuos u objetos presentes, así como excusarse en una falta de iluminación [49]. Por este motivo, la utilización de imágenes del espectro térmico, junto con técnicas avanzadas de procesamiento basadas en redes neuronales convolucionales, se presenta como una alternativa ideal a la situación previamente planteada, y proporciona la base para los sistemas de detección de objetos en escenarios críticos, enfoque adoptado en el presente trabajo.

3 DESARROLLO DEL TFG

En este capítulo se exponen detalladamente las tareas realizadas durante el transcurso del Trabajo de Fin de Grado. En primer lugar, se muestra el área de estudio donde se ha llevado a cabo la adquisición de imágenes, así como las plataformas desde las que fueron tomadas y los objetos de interés para su posterior procesamiento. Seguidamente, se explican los equipos utilizados para la captura de imágenes y, finalmente, se desarrolla el procesamiento de las imágenes obtenidas para el uso de los modelos de *Deep Learning*, desde la preparación y etiquetado hasta la aplicación de ambos modelos, YOLO y Mask R-CNN, con el objetivo de conseguir la detección de objetos automatizada en entornos semejantes al del caso estudiado.

3.1 Área de estudio

Para el presente trabajo el área de estudio seleccionada se localiza en la ría de Pontevedra y, en particular, en las inmediaciones de la Escuela Naval Militar (véase Figura 3.1), situada en el municipio de Marín (Pontevedra). Dicho lugar presenta las características adecuadas para el análisis que se desea realizar en el trabajo, al combinar zonas de tráfico marítimo, áreas costeras, infraestructuras portuarias y condiciones ambientales variables. Asimismo, la oportunidad de adquirir imágenes desde distintas plataformas, tanto fijas como móviles, ha facilitado la obtención de un conjunto de datos variado y realista.



Figura 3.1 Localización de la Escuela Naval Militar en la Ría de Pontevedra (Coordenadas 42° 23' N, 008° 42'W) (Fuente: Google Earth)

3.1.1 Base fija

Para la obtención de imágenes desde plataformas fijas, éstas fueron tomadas en la zona norte dentro del recinto de la Escuela Naval Militar, como puede observarse en la Figura 3.2.



Figura 3.2 Vista general del área de estudio dentro del recinto de la Escuela Naval Militar (Fuente: Google Earth)

En esta área se escogieron tres zonas (véase Figura 3.3) con el objetivo de considerar distintos puntos de vista (Figura 3.4, Figura 3.5, Figura 3.6) y variedad de fondos tras los objetos de interés en estudio.



Figura 3.3 Distribución de las bases fijas escogidas para la toma de imágenes (Fuente: Google Earth)



Figura 3.4 Vista general desde la zona B1 (Elaboración propia)



Figura 3.5 Vista general desde la zona B2 (Elaboración propia)



Figura 3.6 Vista general desde la zona B3 (Elaboración propia)

3.1.2 Base móvil

Para las imágenes adquiridas desde una plataforma móvil, se han llevado a cabo 2 navegaciones a lo largo de la ría de Pontevedra tanto en lanchas de instrucción (véase Figura 3.7), utilizadas en periodos de adiestramiento a flote de los alumnos de la Escuela Naval Militar, como en embarcaciones auxiliares utilizadas para el transporte de personal como la Y-504 (véase Figura 3.8). Con ello se ha conseguido ampliar el área de estudio, así como la cantidad y variedad de objetos de interés, desde los de pequeñas dimensiones, como palangres, hasta pesqueros o cargueros de grandes dimensiones.



Figura 3.7 Lancha de instrucción A-121 'Guardiamarina Barrutia' (Fuente: Facebook 'Escuela Naval Militar')



Figura 3.8 Embarcación auxiliar Y-504 en la Escuela Naval Militar [50]

3.1.3 Objetos de interés

El entorno marítimo de la ría de Pontevedra presenta una elevada diversidad de elementos susceptibles de ser considerados objetos de interés desde el punto de vista de la seguridad, la vigilancia y las operaciones de búsqueda y rescate. Entre ellos se encuentran los siguientes:

- **Embarcaciones de pequeño/medio/gran tamaño:** en la ría es habitual la presencia de embarcaciones de recreo, lanchas neumáticas, pesqueros y embarcaciones auxiliares, así como buques de mayor porte en tránsito o fondeados (véase Figura 3.9).



Figura 3.9 Embarcaciones de gran tamaño en tránsito o fondeados en la ría (Elaboración propia)

- **Objetos flotantes:** debido a la actividad pesquera, el área de estudio incluye la presencia de objetos flotantes como palangres (véase Figura 3.10), boyas, redes u otros elementos a la deriva. Estos objetos, aunque de pequeño tamaño, suponen un riesgo significativo para la

navegación, especialmente para embarcaciones rápidas o de reducido calado, y pueden ser difíciles de detectar mediante sistemas convencionales.



Figura 3.10 Palangre a la deriva en la ría (Elaboración propia)

- **Náufragos:** debido a la especial relevancia de la detección de personas en el medio acuático, como naufragos o individuos en situaciones de emergencia, en este trabajo se consideran escenarios controlados correspondientes a ejercicios de “hombre al agua” realizados por el alumnado de la Escuela Naval Militar, como puede observarse en la Figura 3.11. Este es un escenario crítico en la realidad debido al reducido tamaño del objetivo y a su bajo contraste en el espectro visible.



Figura 3.11 Muñeco de simulación de hombre al agua en ejercicios de lanchas de instrucción (Elaboración propia)

Finalmente, también se contemplan otros elementos presentes en el entorno marítimo, como restos flotantes, embarcaciones semisumergidas, fauna marina y elementos estructurales en zonas próximas a la costa.

3.2 Equipos empleados para la toma de imágenes

3.2.1 Cámara termográfica FLIR E-85

La FLIR E-85 (Figura 3.12) es un equipo profesional de altas prestaciones, diseñado para aplicaciones de inspección, análisis térmico y monitorización avanzada, proporcionando imágenes térmicas con metadatos. Sus principales propiedades pueden verse en la Tabla 3-1.



Figura 3.12 Cámara termográfica FLIR E-85 [51]

Entre las características más relevantes de la FLIR E-85 se encuentra su capacidad para generar imágenes radiométricas, en las que cada píxel contiene información de temperatura en °C. Este aspecto resulta especialmente útil para el procesamiento posterior de las imágenes, ya que permite un análisis más preciso de la información térmica capturada. Además, para facilitar la adaptación a diferentes condiciones de adquisición, la cámara ofrece distintas paletas de color, ajustes de rango térmico y un sistema de autoenfoco láser, garantizando que las imágenes sean siempre lo más nítidas posible [52], como puede observarse en la Figura 3.13.

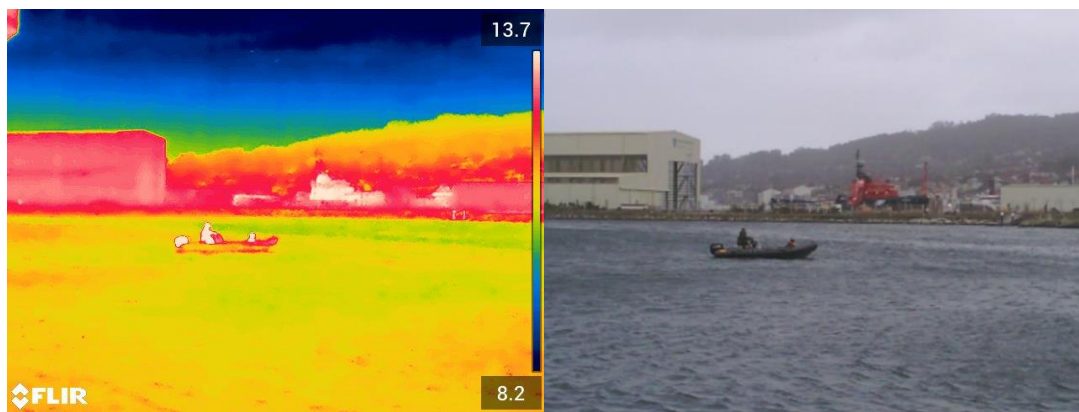


Figura 3.13 Comparación de imágenes térmica y visible realizadas con la FLIR E-85 (Elaboración propia)

Debido a su elevada resolución térmica, se obtiene una mayor capacidad para discriminar detalles térmicos finos y detectar objetos de pequeño tamaño a distancias medias y largas. Asimismo, presenta una alta sensibilidad térmica, lo que permite identificar diferencias de temperatura reducidas entre los elementos presentes en la escena, aspecto especialmente relevante en entornos marítimos donde el contraste térmico puede ser limitado.

Parámetro	Descripción
Tipo de sensor	Microbolómetro no refrigerado
Resolución térmica	384 (ancho) x 288 (altura) píxeles
Resolución visible	5 megapíxeles
Rango espectral	LWIR (7,5-14 μm)
Rango medición de temperatura	-20°C a +1200°C
Frecuencia de imagen	30 Hz
Sensibilidad térmica	< 30 mK

Tabla 3-1 Características principales de la FLIR E-85 [53]

3.2.2 Cámara termográfica FLIR One Edge

La cámara termográfica FLIR One Edge (Figura 3.14) es un dispositivo compacto y portátil perteneciente a la gama de bajo coste de FLIR Systems, diseñado principalmente para su uso acoplado a dispositivos móviles. Su reducido tamaño y peso facilitan su empleo en diferentes escenarios, permitiendo la captura de imágenes térmicas de forma sencilla y rápida. Sus características pueden observarse en la Tabla 3-2.



Figura 3.14 Cámara termográfica FLIR One Edge [54]

Esta cámara térmica se conecta de forma inalámbrica al dispositivo inteligente para poder inspeccionar fácilmente objetivos que están fuera de alcance o ajustarla al teléfono móvil o tablet para que funcione con una sola mano. Al combinar VividIR™ y FLIR MSX® (imagen dinámica multiespectral), puede tomar imágenes térmicas nítidas desde cualquier posición o ángulo con un agarre natural. FLIR Ignite™ permite cargar al instante imágenes y vídeos de la cámara en la nube, donde se pueden editar, organizar, almacenar y compartir datos. Compatible con smartphones y tablets iOS y Android, la FLIR One Edge se puede utilizar sin limitaciones, independientemente del operador, sistema operativo, tamaño o futuras actualizaciones de dispositivos móviles [54].

Parámetro	Descripción
Tipo de sensor	Microbolómetro no refrigerado
Resolución térmica	80x60 píxeles
Resolución visible	640x480 píxeles
Rango espectral	LWIR (8-14 μm)
Rango medición de temperatura	-20°C a +120°C
Frecuencia de imagen	8,7 Hz
Sensibilidad térmica	< 70 mK

Tabla 3-2 Características principales de la FLIR One Edge [55]

En comparación con cámaras termográficas de gama profesional, la FLIR One Edge presenta una resolución térmica y una sensibilidad inferiores, lo que se traduce en un menor nivel de detalle y una capacidad más limitada para discriminar pequeñas diferencias de temperatura. La cámara dispone de una frecuencia de imagen reducida y un sistema de enfoque fijo, características que condicionan la calidad de la imagen térmica obtenida, especialmente en escenas dinámicas o con objetivos a diferentes distancias. No obstante, el dispositivo permite la visualización de imágenes térmicas de forma continua y estable dentro de sus márgenes operativos, como puede observarse en la Figura 3.15. Por otro lado, cabe destacar que el coste asociado a este dispositivo es muy inferior al asociado a cámaras termográficas profesionales o de alta gama como es el caso de la FLIR E-85.

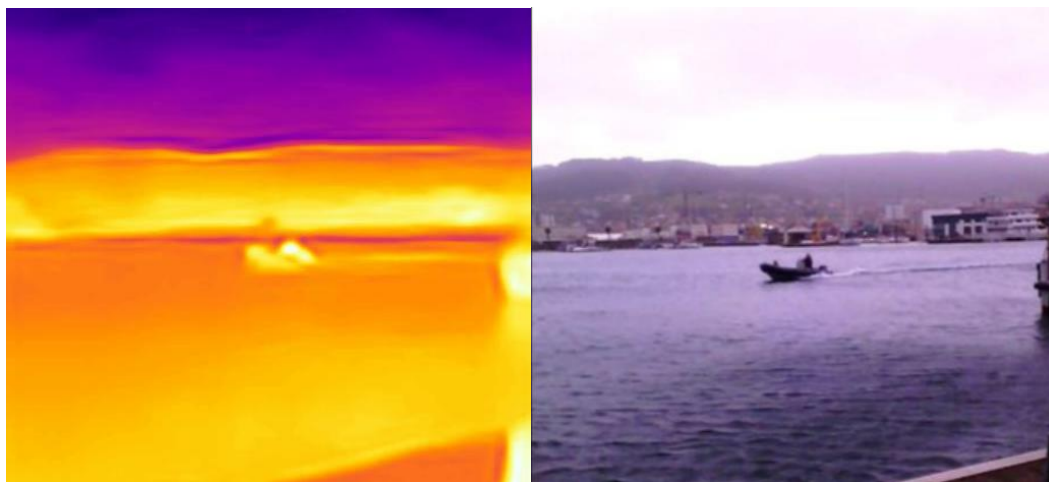


Figura 3.15 Comparación de imágenes térmica y visible realizadas con la FLIR One Edge (Elaboración propia)

3.3 Procesamiento de imágenes

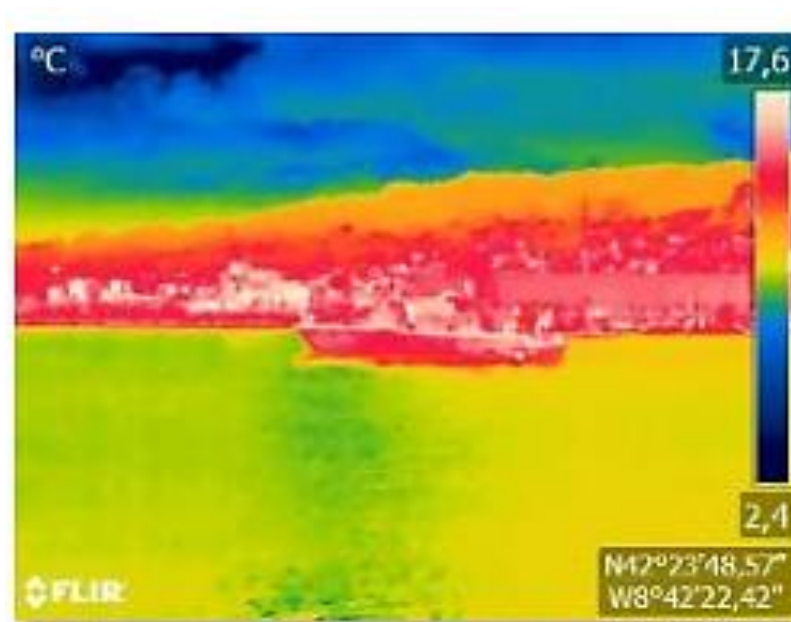
Para comenzar la segunda parte práctica del trabajo, que conlleva previamente el procesamiento de las imágenes una vez adquiridas, se han realizado de forma repetitiva y secuencial tres tareas principales; en primer lugar, la clasificación y nomenclatura de las imágenes para una mejor gestión, diferenciando así sus correspondencias con ambas cámaras, así como la ubicación, el tipo de espectro en el que se han realizado, la fecha de adquisición y el objeto de interés capturado. El espectro visible se emplea como complemento de las imágenes térmicas a la hora de tratar de mejorar el aprendizaje de los modelos DL empleados en este trabajo. Seguidamente se ha realizado el etiquetado de las imágenes mediante una herramienta online de código abierto y, posteriormente, su aplicación como fuente de datos de entrada para el aprendizaje de los modelos de *Deep Learning*.

3.3.1 Clasificación y etiquetado

El paso previo al etiquetado de las imágenes es su clasificación y nomenclatura, para lo cual se ha utilizado la estructura ‘ESPECTRO_CÁMARA_BASE_FECHA_OBJETO’, como puede observarse en la Figura 3.16, con el objetivo de agilizar la gestión e identificación de las imágenes. Los parámetros considerados para la nomenclatura son los siguientes:

- ESPECTRO: será ‘IR’ (infrarrojo) o ‘VIS’ (visible) según corresponda.
- CÁMARA: será ‘E85’ si se ha realizado con la cámara de alta gama FLIR E85, o ‘ONE’ si se trata de la cámara de bajo coste FLIR One Edge.
- BASE: hace referencia a las tres bases fijas y las dos bases móviles:
 - B1: fija, detrás del edificio de investigación con vistas a la ría.
 - B2: fija, muelle Almirante Vierna con vistas a la dársena.
 - B3: fija, faro de torpedos.
 - BM: móvil, lancha de instrucción.
 - BM1: móvil, embarcación Y-504.
- FECHA: día de adquisición de la imagen siguiendo el formato DDMMAA.

- OBJETO: inicial del tipo de objeto de interés detectado:
 - A: ave.
 - B: barco.
 - H: humanos.
 - L: lancha de instrucción.
 - M: marca de señalización marítima.
 - N: naufrago.
 - P: palangre.
 - R: rhib neumática.
 - T: tronco.
 - V: velero.



IR_E85_BM_170226_L

Figura 3.16 Ejemplo de nomenclatura de imagen térmica (Elaboración propia)

Para el proceso de etiquetado se ha utilizado la herramienta de código abierto VGG Image Annotator (VIA), un programa de anotación manual sencillo e independiente para imágenes, audio y vídeo. Se trata de una extensión HTML que permite cargar las imágenes térmicas y visibles, crear atributos y asignarlos a regiones especificadas en las imágenes [56].

Los pasos seguidos para el etiquetado de las imágenes fueron los siguientes:

- **Carga de imágenes:** cómo se puede observar en la Figura 3.17, en la pestaña ‘Project’ de la barra superior de la interfaz de VGG, aparece la opción ‘Add local files’ para añadir las imágenes que se deseen desde la carpeta del ordenador, ya sea formato JPG, PNG o GIF. Se habrá creado entonces un proyecto con un nombre editable, que será el exportado posteriormente para su implementación en el código de los modelos. Cabe destacar que las imágenes térmicas obtenidas con la cámara FLIR E-85 se emplean en formato imagen convencional, y no en formato radiométrico que contenga la información de temperatura en grados Celsius para cada píxel. Esto implica que durante el proceso se pierde la información radiométrica original del sensor. No obstante, dado que la cámara FLIR One Edge no dispone de información radiométrica por píxel, en este trabajo se ha optado por utilizar ambas cámaras bajo las mismas condiciones de entrada, empleando únicamente imágenes térmicas en formato visual. De este modo, se busca evaluar el rendimiento de los modelos de Deep

Learning cuando trabajan con datos térmicos sin información radiométrica completa, permitiendo analizar hasta qué punto es posible obtener resultados útiles a partir de información térmica menos precisa.

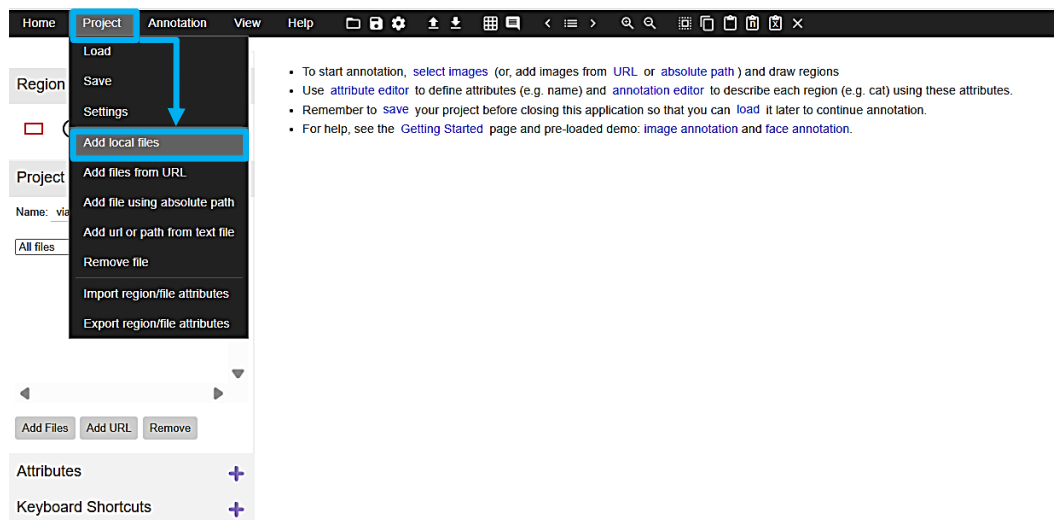


Figura 3.17 Primer paso de carga de imágenes en la página principal de la herramienta VGG (Elaboración propia)

- **Creación de atributos:** para facilitar la identificación de los objetos a detectar por los modelos DL, se crean a continuación una serie de propiedades o características denominadas atributos, las cuales se le asignarán a la región que delimita el objeto que se desea detectar (véase Figura 3.18). Para ello se accede a la pestaña 'Attributes' del menú lateral izquierdo, se añade el atributo en cuestión, en este caso 'Objeto', y en la opción 'type' se introduce el valor de 'dropdown', el cual permite introducir otras características del atributo para aclarar la clasificación del objeto. Para el enfoque de este trabajo, se escogieron cuatro clasificaciones: 'GrandeLejos', 'GrandeCerca', 'PequenoLejos' y 'PequenoCerca'. La letra ñ se introduce sin virgulilla para evitar posibles errores durante el desarrollo del código.

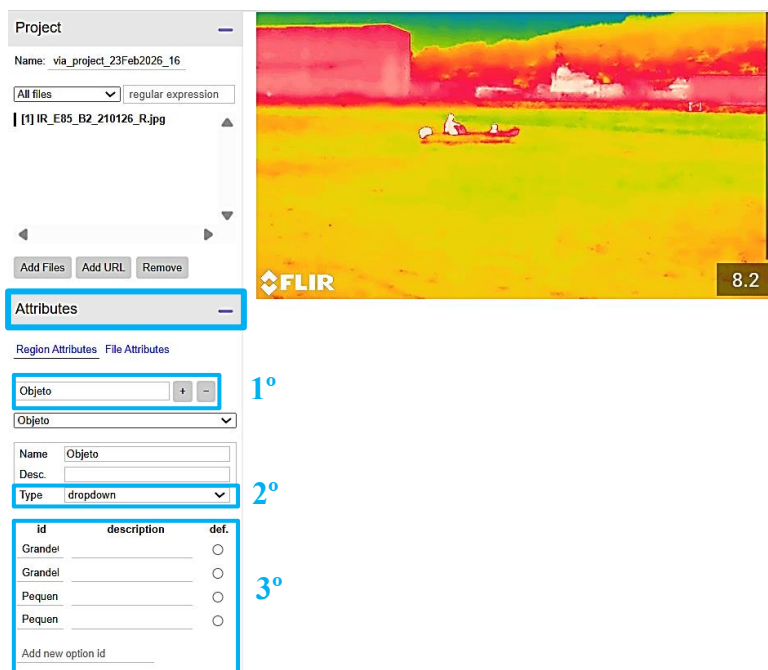


Figura 3.18 Ejemplo del procedimiento de creación de atributos (Elaboración propia)

- **Identificación y clasificación:** una vez definidos los atributos necesarios, se procede a delimitar la región en la que se encuentra el objeto que se quiere detectar y clasificar. Accediendo a la pestaña ‘Region shape’, se muestra una variedad de formas de dibujo, de entre las cuales se escoge la ‘Rectangular’ para el modelo YOLO o la ‘Polygonal’ para el modelo Mask R-CNN, según corresponda. Tras el trazado de la región, aparece una lista desplegable en su interior que permite seleccionar el atributo correspondiente de los planteados en el punto anterior, como puede observarse en la Figura 3.19.

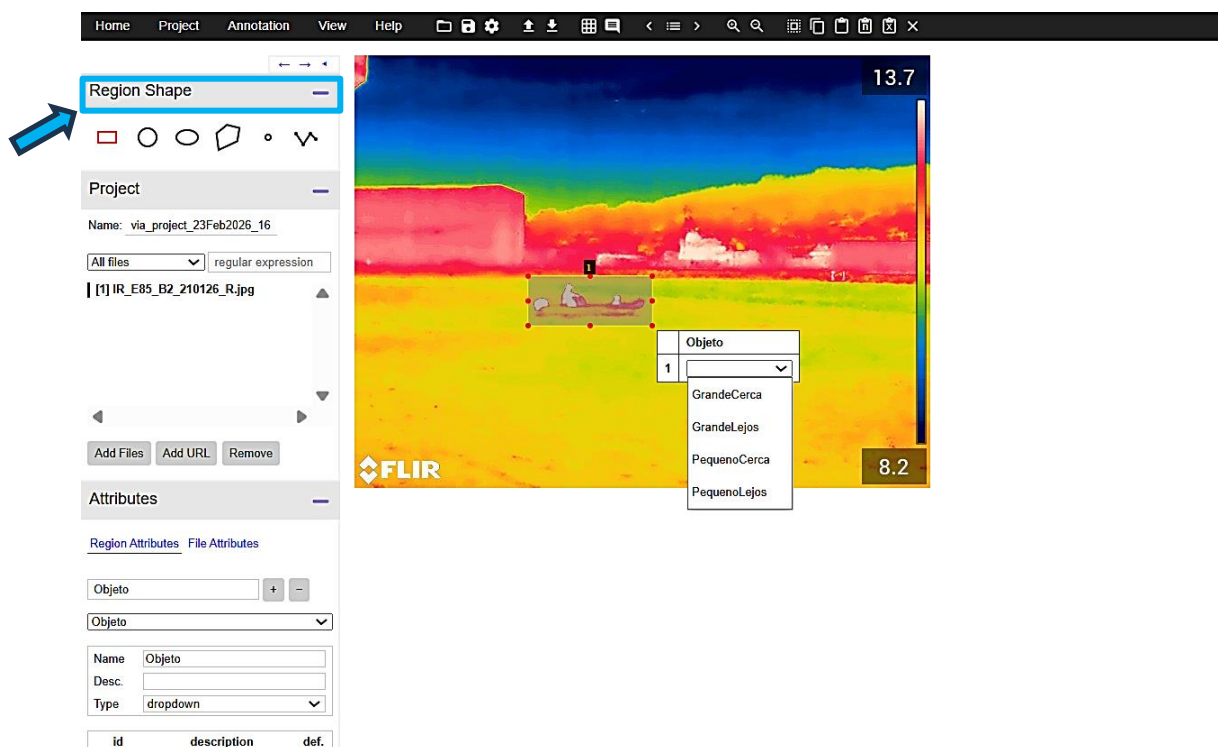


Figura 3.19 Procedimiento de trazado de la región y opciones de clasificación (Elaboración propia)

- **Almacenamiento:** finalmente, la imagen etiquetada se exporta como archivo “.json” desde la pestaña superior ‘Annotation’. Este archivo, el cual será clave para el modelo DL, contiene un código que muestra el nombre de la imagen, su tamaño y las regiones trazadas definiendo sus coordenadas y los atributos aplicados (véase Figura 3.20).

Nombre de la imagen

```
{
  "IR_E85_B2_210126_R.jpg265653": {
    "filename": "IR_E85_B2_210126_R.jpg",
    "size": 265653,
    "regions": [
      {
        "shape_attributes": {
          "name": "rect",
          "x": 193,
          "y": 245,
          "width": 138,
          "height": 55
        },
        "region_attributes": {
          "Objeto": "GrandeCerca"
        },
        "file_attributes": {}
      }
    ]
  }
}
```

Atributos

Forma de región y coordenadas

Figura 3.20 Código del archivo .json exportado con las características del etiquetado de la imagen (Elaboración propia)

3.3.2 Ejecución de los modelos Deep Learning

Con el conjunto de datos preparado y etiquetado, se procede a la implementación y evaluación de las diferentes arquitecturas de detección de objetos basadas en *Deep Learning*. El objetivo de esta fase es analizar el comportamiento de modelos representativos de detección de una etapa y de dos etapas sobre imágenes del espectro infrarrojo y visible, evaluando su capacidad para identificar y localizar los objetos de interés definidos en el apartado anterior. Para ello, se seleccionan dos arquitecturas ampliamente empleadas en la literatura científica: YOLOv11 y YOLOv26, versiones del modelo de

detección de una sola etapa siendo la v26 la versión actual lanzada a inicios de año, y Mask R-CNN, como modelo de dos etapas con capacidad de segmentación.

Para la ejecución y prueba de los modelos se ha empleado la plataforma Google Colab, que permite el uso de entornos de programación en la nube con acceso a recursos de aceleración por GPU, facilitando el aprendizaje y testeo de redes neuronales sin necesidad de disponer de infraestructura local de alto rendimiento [57]. En los siguientes subapartados se describe de forma detallada el procedimiento seguido para la implementación de cada modelo, incluyendo la configuración utilizada, los parámetros de aprendizaje y la estructura del conjunto de datos. El entorno de ejecución y los códigos empleados de ambos modelos, basados en el lenguaje de programación Python, se pueden consultar en el Anexo III: Código YOLO v11 y v26 y en el Anexo IV: Código Mask R-CNN.

3.3.2.1 YOLO

La elección de esta arquitectura se fundamenta en su elevada eficiencia computacional y su capacidad de detección en tiempo real, lo que la convierte en una referencia actual en tareas de detección automática de objetos. El funcionamiento base (arquitectura) del modelo YOLO, la cual se observa en la Figura 3.21, se divide en varias etapas [58], [59]:

- **Input and Backbone network (extracción de características):** la imagen se introduce en el modelo, el cual emplea una arquitectura convolucional profunda como red de extracción de características (*backbone*), incluyéndolas en mapas (*feature maps*) que contienen información espacial y semántica relevante de la imagen. La *backbone*, en este caso *CSPNet*, incorpora capas convolucionales, funciones de activación no lineales y bloques residuales optimizados para capturar patrones jerárquicos, desde bordes y contornos simples hasta representaciones más abstractas.
- **Neck network (agregación de características):** los mapas generados pasan a una etapa intermedia denominada *neck*, cuya función es combinar información procedente de diferentes escalas. Esta etapa permite mejorar la detección de objetos de distintos tamaños, fusionando características profundas (alto nivel semántico) con características más superficiales (mayor resolución espacial). El resultado es una representación enriquecida que facilita la detección tanto de objetos grandes como pequeños dentro de la misma imagen.
- **Detection Head (predicción):** en esta fase la red toma los mapas de características agregados y realiza las predicciones finales. Para cada celda en la imagen de entrada (que YOLO divide internamente), la *head* predice:
 - Clases: las probabilidades de que cada clase de objeto esté presente.
 - Coordenadas del *Bounding Box*: el centro (x, y), el ancho (w) y alto (h) del cuadro delimitador.
 - Puntuación de confianza (*Objectness Score*): la probabilidad de que una clase de objeto exista en ese cuadro.
- **Training phase (cálculo de pérdidas y optimización):** durante el aprendizaje, el modelo compara sus predicciones con las anotaciones reales del conjunto de datos. Se calculan diferentes componentes de pérdida:
 - Pérdida de localización (ajuste de *bounding boxes*).
 - Pérdida de clasificación.
 - Pérdida de confianza (*objectness*).

Mediante retropropagación (*back propagation*), los pesos de la red se ajustan iterativamente durante un número determinado de *epochs* durante el aprendizaje, optimizando progresivamente el rendimiento del modelo.

- **Non-Maximum Supression (NMS):** durante esta fase de inferencia, el modelo puede generar múltiples predicciones solapadas para un mismo objeto. *NMS* se encarga de eliminar los cuadros delimitadores redundantes, conservando únicamente aquel con la puntuación de confianza más alta para cada objeto detectado.

- **Output (Salida de objetos detectados):** se presentan los cuadros delimitadores finales, las etiquetas de clase asociadas y las puntuaciones de confianza, permitiendo identificar y clasificar automáticamente los objetos de interés presentes en cada escena.

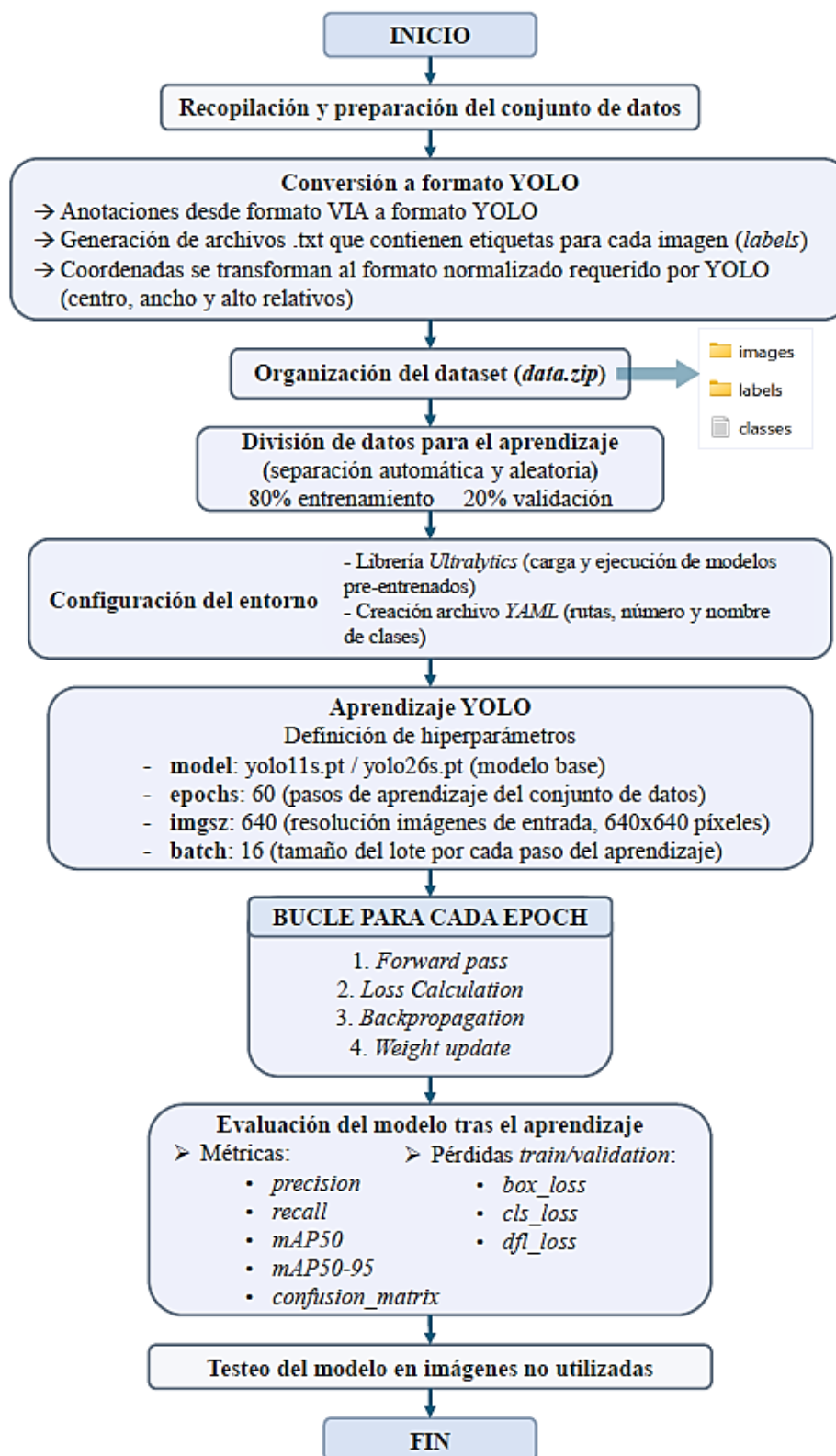


Figura 3.21 Esquema general del funcionamiento del modelo YOLO empleado para la detección de objetos (Elaboración propia)

El proceso descrito comienza con la preparación del conjunto de datos, que incluye la recopilación de las imágenes térmicas y su etiquetado mediante VGG. Las anotaciones generadas por VIA se almacenan inicialmente en formato JSON, donde cada objeto se define mediante un rectángulo caracterizado por su posición (desde la esquina superior izquierda) y dimensiones (alto y ancho) dentro de la imagen. Posteriormente, estas anotaciones se convierten al formato requerido por YOLO mediante un script de transformación que calcula el identificador de clase (*class_id*) y las coordenadas normalizadas del centro del objeto junto con el ancho y alto del cuadro delimitador. De este modo, cada objeto queda representado mediante los parámetros *class_id*, *center_x*, *center_y*, *width* y *height*, todos ellos normalizados respecto al tamaño de la imagen, siendo 0 el valor mínimo y 1 el máximo.

Una vez preparado el conjunto de datos, se configura el proceso de aprendizaje utilizando un modelo preentrenado, que ha sido previamente entrenado sobre el conjunto de datos *COCO*. Este enfoque permite aplicar *transfer learning*, reutilizando las características generales aprendidas por el modelo en tareas de detección de objetos y adaptándolas posteriormente al conjunto de imágenes específico de este trabajo. El aprendizaje del modelo se realiza mediante un proceso iterativo basado en *mini-batch gradient descent*, en el cual los pesos de la red neuronal se actualizan utilizando pequeños subconjuntos de imágenes del conjunto de entrenamiento. Este método constituye un compromiso entre el descenso de gradiente clásico y el descenso estocástico, permitiendo una estimación más estable del gradiente y una mayor eficiencia computacional al aprovechar la paralelización en GPU. Durante este proceso intervienen diversos hiperparámetros, que influyen directamente en la velocidad de entrenamiento y en la capacidad del modelo para generalizar a nuevos datos, entre los que destacan la tasa de aprendizaje inicial (*lr0*) establecida en 0,01, que controla la magnitud de las actualizaciones de los pesos del modelo, el *momentum* de 0,937, que introduce inercia en la dirección del gradiente para acelerar la convergencia del entrenamiento, y el *weight_decay*, una técnica de regularización para evitar el sobreajuste del modelo, con un valor de 0,0005 [60].

A lo largo del aprendizaje, el modelo intenta minimizar diferentes funciones de pérdida que evalúan su rendimiento. Entre ellas se encuentran *box_loss*, que mide la precisión con la que el modelo localiza las cajas delimitadoras de los objetos; *cls_loss*, que evalúa la capacidad del modelo para clasificar correctamente cada objeto; y *dfl_loss*, que refina la predicción de las coordenadas de las cajas delimitadoras mediante un enfoque basado en distribuciones probabilísticas. La evolución de estas pérdidas permite analizar el proceso de aprendizaje del modelo y detectar posibles problemas de sobreajuste o falta de convergencia.

Finalmente, el rendimiento del modelo se evalúa mediante métricas estándar de detección de objetos. La precisión (*precision*) mide la proporción de detecciones correctas respecto al total de detecciones realizadas por el modelo, es decir, indica cuántas de las predicciones positivas son realmente correctas. Se calcula como el cociente entre verdaderos positivos y la suma de verdaderos positivos y falsos positivos. El *recall* mide la capacidad del modelo para detectar todos los objetos presentes en la imagen, representando la proporción de verdaderos positivos respecto al total de objetos reales. Se obtiene como el cociente entre verdaderos positivos y la suma de verdaderos positivos y falsos negativos. Por último, el *mAP* (*mean Average Precision*) es una métrica global que combina *precision* y *recall*, calculando el área bajo la curva *precision-recall* para cada clase y promediando los resultados. En particular, *mAP50* evalúa las detecciones considerando un umbral de intersección sobre unión (*IoU*) del 50%, mientras que *mAP50-95* promedia este valor para distintos umbrales entre 0.5 y 0.95, proporcionando una medida más estricta del rendimiento del modelo. Estas métricas se calculan sobre el conjunto de validación y constituyen un indicador del comportamiento esperado del modelo cuando se aplica a nuevas imágenes no utilizadas durante el aprendizaje, en la denominada fase de testeo, ya que los pesos durante el aprendizaje se actualizan sólo con el conjunto de imágenes de entrenamiento.

La principal diferencia entre YOLO11s (lanzado en 2024) y YOLOv26 (lanzado en enero de 2026), radica en que este último ha sido diseñado siguiendo una perspectiva ‘*edge-first*’, es decir, orientado a su ejecución en dispositivos de computación en el borde. Este enfoque prioriza la eficiencia computacional y la velocidad de inferencia, reduciendo la complejidad del modelo y eliminando etapas

de post-procesamiento tradicionales como la NMS. De este modo, el propio modelo gestiona la eliminación de cuadros delimitadores superpuestos durante la inferencia, lo que reduce la variabilidad de la latencia y simplifica su despliegue en sistemas con recursos limitados [61]. La familia de modelos YOLO se ofrece en cinco tamaños: *nano (n)*, *small (s)*, *medium (m)*, *large (l)* y *extra large (x)*. A medida que aumenta el tamaño del modelo, también lo hace su capacidad de representación y, por tanto, su precisión, aunque a costa de un mayor tiempo de procesamiento y mayores requisitos computacionales. Por este motivo, para el presente caso de estudio, centrado en aplicaciones de seguridad marítima, se ha seleccionado el modelo “s”, priorizando la rapidez de detección frente a una posible ganancia marginal en precisión.

3.3.2.2 Mask R-CNN

En esta segunda etapa del proceso experimental, se ha empleado el modelo Mask R-CNN, el cual combina en una misma arquitectura tres tareas principales: detección, clasificación y segmentación de objetos. A diferencia de los modelos de detección que únicamente proporcionan cajas delimitadoras (*bounding boxes*), Mask R-CNN permite además generar una máscara de segmentación para cada objeto detectado, identificando de forma precisa los píxeles que lo componen dentro de la imagen.

El modelo (cuyo diagrama de flujo se observa en la Figura 3.22) se basa en una arquitectura de dos etapas (*two-stage detector*). En la primera fase se generan las regiones de interés dentro de la imagen que pueden contener objetos, mientras que en la segunda fase estas regiones son analizadas con mayor detalle para determinar su clase, ajustar con precisión las cajas delimitadoras y generar las máscaras de segmentación correspondientes [62]. A continuación, se describen las fases que componen el proceso de funcionamiento (arquitectura) del modelo empleado [63]:

- **Input (entrada de la imagen):** se introducen las imágenes en el sistema, de modo que se transforman en un formato tensorial adecuado para su procesamiento por la red neuronal.
- **Backbone Network (extracción de características):** de igual forma que para el modelo YOLO, las imágenes de entrada son procesadas por una red neuronal convolucional, en este caso ResNet, combinados con una *Feature Pyramid Network (FPN)*. En esta etapa la red analiza la imagen aplicando filtros convolucionales que permiten identificar patrones básicos como bordes, contornos o texturas. Como resultado se generan los mapas de características (*feature maps*), que contienen la información visual de la imagen.
- **Region Proposal Network (generación de regiones candidatas):** a partir de los mapas obtenidos, esta red tiene como objetivo identificar las posibles regiones de la imagen donde pueda existir un objeto. Para ello se utilizan *anchor boxes*, cajas delimitadoras de diferentes tamaños y proporciones que se desplazan sobre el mapa de características. La RPN evalúa cada una de estas cajas y genera una serie de propuestas de regiones (*Region Proposals*) que presentan una alta probabilidad de contener un objeto con la estimación de la clase.
- **ROI Align (selección y alineamiento de regiones):** este procedimiento permite ajustar y alinear con precisión las regiones seleccionadas por el RPN con los mapas de características salientes del *Backbone Network*, preservando la información espacial original de la imagen. Este paso es fundamental para mejorar la precisión en la localización de los objetos, especialmente en tareas de segmentación.
- **Clasificación y refinamiento de cajas delimitadoras:** una vez alineadas las regiones de interés, se introducen en una serie de capas de la red que realizan, por una parte, la clasificación del objeto, determinando a qué categoría pertenece cada región detectada; y por otra parte, el refinamiento de la caja delimitadora, ajustando con mayor precisión las coordenadas alrededor del objeto.
- **Mask Prediction (generación de máscaras de segmentación):** la diferencia con otros modelos de predicción es la incorporación de esta tercera rama de procesamiento, destinada a la segmentación de instancias. En esta etapa, el modelo genera una máscara binaria para cada objeto detectado, indicando exactamente qué píxeles de la imagen pertenecen al objeto.

Esto permite, además de localizar el objeto mediante la caja delimitadora, identificar su forma y extensión dentro de la imagen.

- **Output (salida del modelo):** finalmente, el modelo produce como resultado tres tipos de información para cada objeto detectado: la clase del objeto (*class label*), la caja delimitadora (*bounding box*) y la máscara de segmentación (*object mask*). Estos resultados se superponen sobre la imagen original, permitiendo visualizar la localización y clasificación de los objetos.

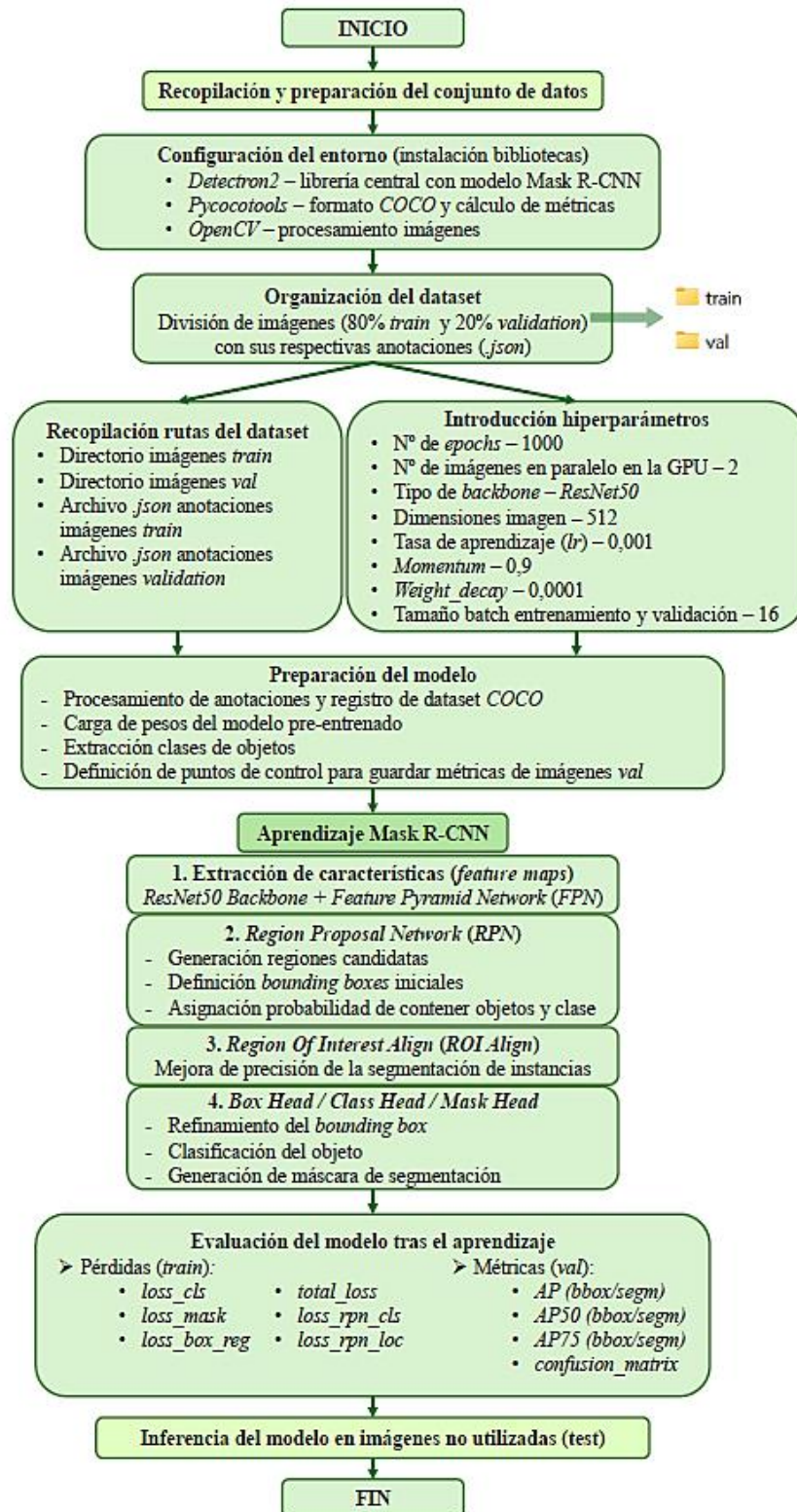


Figura 3.22 Esquema general del funcionamiento del modelo Mask R-CNN empleado para la detección de objetos (Elaboración propia)

El proceso de funcionamiento del modelo Mask R-CNN empleado en este trabajo comienza, al igual que en el modelo YOLO, con la recopilación y preparación del conjunto de datos que constituye la base para el aprendizaje del modelo. Las anotaciones generadas inicialmente en formato JSON se convierten al formato requerido por Mask R-CNN, basado en el estándar *COCO*, donde se incluyen las coordenadas de las *bounding boxes*, la clase asociada a cada objeto y las máscaras de segmentación correspondientes.

A diferencia del caso de YOLO, la organización de los datos se realiza mediante la división manual previa de las imágenes en dos conjuntos principales (80/20). Por un lado, el directorio destinado al *training*, que contiene las imágenes de entrenamiento junto con su archivo de anotaciones en formato JSON, y por otro lado el conjunto de *validation*, compuesto igualmente por las imágenes de validación y su correspondiente archivo de anotaciones. Posteriormente se configura el entorno de ejecución del código, instalando las librerías necesarias para el funcionamiento del modelo y las herramientas empleadas para la evaluación de métricas y el procesamiento de imágenes. En esta fase también se cargan las rutas correspondientes a las carpetas que contienen las imágenes y los archivos de anotaciones utilizados durante el entrenamiento y la validación. Una vez configurado el entorno, se introducen los hiperparámetros que controlan el proceso de aprendizaje, como el número de clases que deben detectarse, el tamaño del *batch* de entrenamiento, el número de iteraciones y el valor de la tasa de aprendizaje (*learning rate*). Asimismo, se emplea un modelo preentrenado sobre el dataset *COCO*, lo que permite aplicar técnicas de *transfer learning*. Este enfoque aprovecha las características aprendidas por la red durante su entrenamiento previo en grandes conjuntos de datos, reduciendo el tiempo de entrenamiento necesario. Como se aprecia, los valores de los hiperparámetros son similares al caso del YOLO a excepción del número de *epochs*, dada la mayor complejidad de la arquitectura Mask R-CNN.

Durante la fase de aprendizaje, cada imagen de entrada es procesada inicialmente por la red *backbone*, que genera mapas de características (*feature maps*) a partir de la información visual de la imagen. Estos mapas de características son posteriormente utilizados por la *Region Proposal Network* (RPN), cuya función es generar distintas regiones candidatas que podrían contener objetos dentro de la imagen. Cada una de estas regiones se representa mediante un cuadro delimitador preliminar y una probabilidad asociada de contener un objeto. Debido a que estas regiones candidatas pueden solaparse entre sí, se aplica el algoritmo *Non-Maximum Suppression* (NMS) dentro de la RPN para eliminar las propuestas redundantes y conservar únicamente aquellas con mayor probabilidad de contener un objeto. Las regiones propuestas por la RPN se procesan posteriormente mediante la operación *ROI Align*, que permite ajustar con precisión las regiones de interés sobre los mapas de características generados por la red. Este procedimiento evita pérdidas de información espacial y garantiza que las características extraídas correspondan exactamente con la posición del objeto dentro de la imagen.

A partir de cada región de interés, el modelo ejecuta la denominada *Mask R-CNN Head*, que constituye la etapa final de predicción del modelo. En esta fase se realizan tres tareas principales de forma paralela: la clasificación del objeto detectado, el refinamiento de las coordenadas del cuadro delimitador y la generación de una máscara de segmentación que delimita con precisión la región ocupada por el objeto dentro de la imagen. Esta máscara permite obtener una segmentación a nivel de píxel, lo que representa una de las principales ventajas de Mask R-CNN frente a otros modelos. Durante el aprendizaje del modelo, se calculan diferentes funciones de pérdida (*loss functions*) asociadas a la clasificación de los objetos, la localización de las cajas delimitadoras y la generación de las máscaras de segmentación. La minimización de estas funciones de pérdida permite ajustar progresivamente los pesos de la red neuronal para mejorar la precisión del modelo.

Finalmente, el rendimiento del modelo se evalúa mediante métricas estándar de detección de objetos en el conjunto de datos de validación. El *AP* (*Average Precision*) mide la precisión media del modelo a lo largo de distintos niveles de *recall*, calculando el área bajo la curva *precision-recall*. Esta métrica tiene en cuenta tanto la capacidad de detección como la calidad de las predicciones realizadas. El *AP50* evalúa el rendimiento del modelo considerando un umbral de *Intersection over Union* (IoU) del 50%, lo que implica que una detección se considera correcta si el solapamiento entre la predicción y la anotación real supera dicho valor. Esta métrica es menos estricta y refleja la capacidad general de detección. El

AP75 utiliza un umbral de IoU del 75%, siendo una métrica más exigente, ya que requiere una mayor precisión en la localización de los objetos. Por tanto, permite evaluar con mayor detalle la calidad de las predicciones.

Además, Mask R-CNN evalúa el rendimiento de forma diferenciada para *bounding boxes (bbox)* y *segmentación (segm)*. En el caso de *bbox*, las métricas analizan la precisión en la localización de los cuadros delimitadores, mientras que en *segm* se evalúa la capacidad del modelo para generar máscaras de segmentación precisas a nivel de píxel. Finalmente, el uso de distintos umbrales de IoU y la separación entre *bbox* y *segm* permite obtener una evaluación más completa del comportamiento del modelo, considerando tanto la detección como la segmentación de los objetos

Una vez finalizado el proceso de aprendizaje, el modelo se aplica en la fase de inferencia sobre imágenes que no han sido utilizadas durante el aprendizaje, permitiendo obtener como salida las cajas delimitadoras, las etiquetas de clase y las máscaras de segmentación correspondientes a los objetos detectados en esta fase de testeo.

4 RESULTADOS Y DISCUSIÓN

Una vez completadas las fases de adquisición de datos, procesamiento de imágenes y aprendizaje de los modelos de detección, en el presente capítulo se exponen y analizan los resultados obtenidos. El objetivo principal es evaluar el rendimiento de los modelos implementados, YOLO y Mask R-CNN, en la detección automática de los objetos de interés definidos en el área de estudio. Para ello, se contemplan tres escenarios diferenciados por el tipo de imágenes utilizados en ellos: un primer escenario con imágenes térmicas de la FLIR E-85, el segundo escenario con las imágenes térmicas de la FLIR E-85 y de la FLIR One Edge y el tercer y último escenario con las incluidas anteriormente junto con imágenes visibles.

En primer lugar, se presentan los resultados cuantitativos mediante la aplicación de ambos modelos en los distintos escenarios. Los resultados derivados del proceso de aprendizaje incluyen métricas de evaluación, explicadas en el apartado previo, que permiten valorar de forma objetiva la capacidad de cada modelo para localizar y clasificar correctamente los objetos presentes en las imágenes térmicas. Asimismo, se analizan las curvas de aprendizaje, es decir, la evolución de las pérdidas con el fin de estudiar la convergencia y estabilidad de cada modelo.

Hay que destacar que para el entrenamiento y validación del modelo YOLO se ha utilizado un conjunto de imágenes etiquetadas que se divide automáticamente y de forma aleatoria en los subconjuntos de *train* y *validation*, mientras que en Mask R-CNN la distribución se realiza previamente de forma manual. Asimismo, durante la preparación del conjunto de datos se ha procurado que el número de instancias por clase fuese lo más equilibrado posible, con el objetivo de evitar problemas asociados al desbalanceo de clases durante el proceso de aprendizaje.

Posteriormente, se realiza un análisis cualitativo de las detecciones obtenidas, examinando ejemplos representativos de predicciones en diferentes imágenes tanto de validación como de testeo. Este análisis permite identificar fortalezas, posibles errores de clasificación, falsas detecciones y limitaciones asociadas tanto al modelo como a las características propias de las imágenes empleadas.

Finalmente, se lleva a cabo una comparación entre ambos enfoques de detección, valorando no solo su rendimiento en términos de precisión, sino también aspectos como la robustez, la capacidad de generalización y la adecuación al tipo de problema planteado. Esta discusión permitirá extraer conclusiones fundamentadas sobre la viabilidad de la aplicación de técnicas de *Deep Learning* en el procesamiento automático de imágenes térmicas dentro del ámbito de la seguridad marítima.

4.1 Resultados en el Escenario 1

En este primer escenario se muestran los resultados obtenidos mediante la aplicación de los modelos de *Deep Learning* sobre las imágenes térmicas capturadas solamente con la cámara FLIR E-85. En

primer lugar se explican los resultados obtenidos con las versiones 11 y 26 del modelo YOLO y, en segundo lugar, los correspondientes al modelo Mask R-CNN. El conjunto de datos para el Escenario 1, constituido por 120 imágenes, permitirá evaluar el comportamiento de los modelos utilizando imágenes con mayor resolución y calidad térmica.

YOLO: para las versiones 11 y 26 del modelo, el conjunto de datos se ha dividido de la siguiente forma:

- 100 imágenes para el aprendizaje de las cuales, 80 (80%) son para entrenamiento y 20 (20%) para validación, asignadas de forma aleatoria por el código.
- 20 imágenes para testeo no utilizadas previamente en el aprendizaje, que serán fundamentales para conocer la eficacia final del modelo.

Durante la ejecución del modelo se monitoriza la evolución de las funciones de pérdida (*loss*) y métricas de evaluación, que permiten analizar el comportamiento del aprendizaje al largo de los distintos *epochs*. Tal como se observa en las gráficas de pérdidas (véase Figura 4.1), las tres funciones del conjunto de datos de entrenamiento presentan una tendencia decreciente y que tiende a 0 a lo largo del proceso de aprendizaje, lo que indica que el modelo va ajustando progresivamente sus parámetros y mejorando su capacidad para localizar y clasificar los objetos. Se puede comprobar así que con un mayor número de *epochs*, esta tendencia tiende a estabilizarse. Sin embargo, se puede observar un ligero sobreajuste en las pérdidas del conjunto de validación, denominado *overfitting*. A pesar de ello, se considera un resultado aceptable al ser menor que 2. Este comportamiento es habitual en modelos de *Deep Learning*, viéndose acentuado en este trabajo debido principalmente al pequeño tamaño del conjunto de datos de entrenamiento y validación y a la variabilidad existente en las condiciones de las imágenes, factores que pueden provocar una mayor dispersión en las métricas de validación.

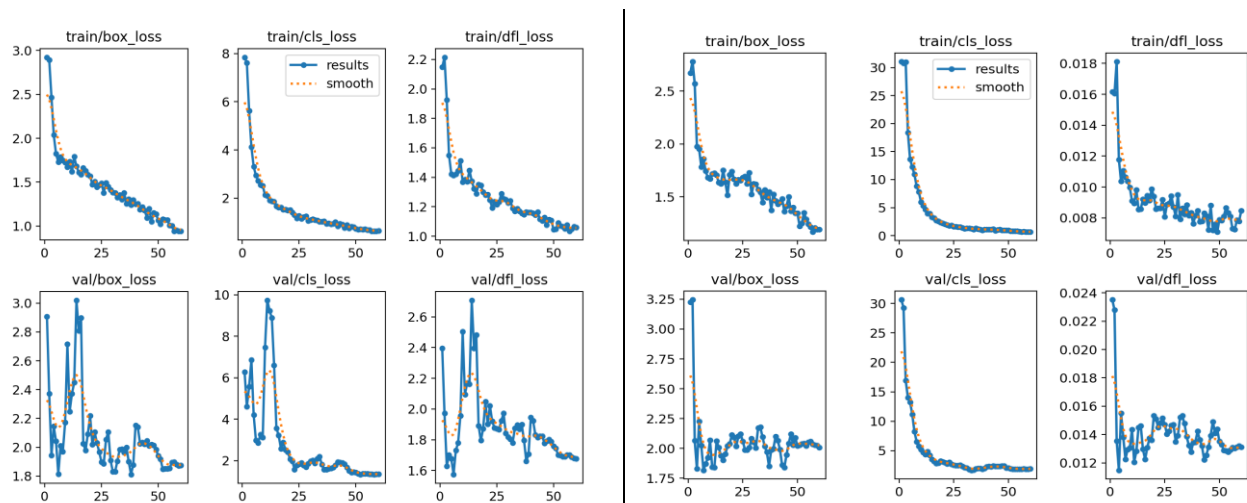


Figura 4.1 Resultados gráficos de las funciones de pérdida (*loss*) de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) en el Escenario 1 (Elaboración propia)

De forma paralela, se analizan las principales métricas de evaluación en el conjunto de validación (véase Figura 4.2). La evolución de estas métricas muestra una mejora progresiva durante el aprendizaje, alcanzando valores que reflejan una capacidad razonable del modelo. En lo relativo a la '*Precision*', ambos modelos muestran valores elevados (igual o superior al 80%) que indican la correspondencia entre el número de aciertos y las detecciones realizadas. En este aspecto, se puede observar una ligera mejora del modelo YOLOv26, superando dicho valor, lo que sugiere una mayor capacidad para reducir falsas detecciones en comparación con la versión 11. En cuanto al '*Recall*', ambos modelos presentan valores similares en torno al 60%.

Tal y como se puede observar en la misma imagen, la métrica *mAP50*, que evalúa la precisión media de detección considerando un umbral de intersección sobre unión del 50%, presenta valores altos en

ambas situaciones (en torno al 70%), mientras que la mAP_{50-95} refleja valores más bajos cercanos al 40%, debido a ser un criterio más exigente evaluando la precisión a umbrales IoU (*Intersection over Union*) superiores.

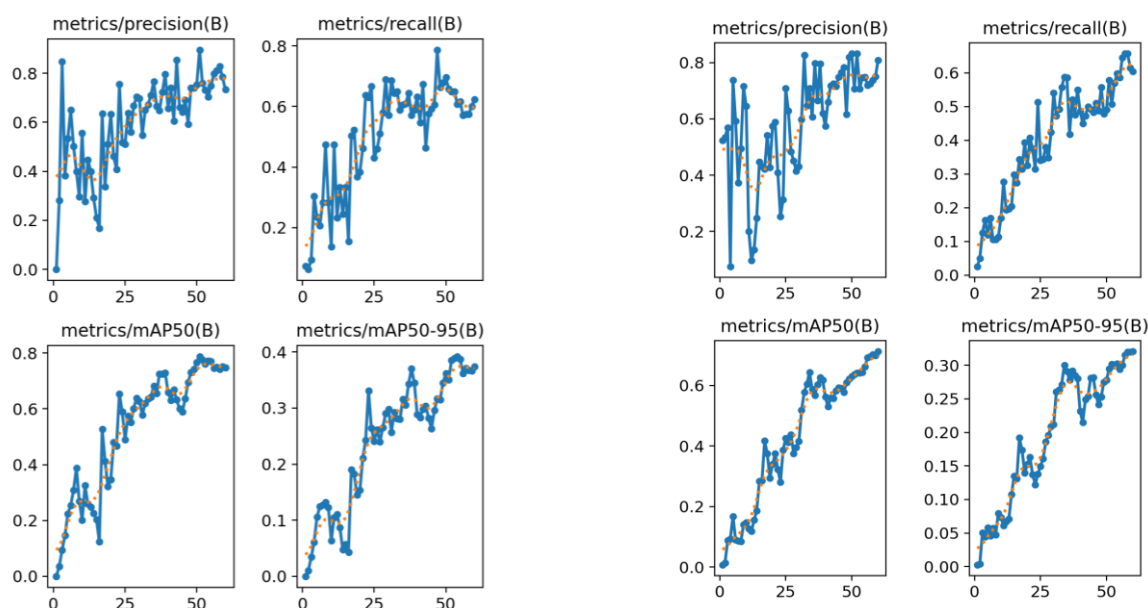


Figura 4.2 Resultado de las métricas de evaluación de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) en el Escenario 1 (Elaboración propia)

Con el objetivo de analizar de forma más detallada el comportamiento del modelo, se generan las matrices de confusión, que permiten visualizar la relación entre las clases reales (*True*) y las clases predichas (*Predicted*) por el modelo, como puede observarse en la Figura 4.3. La matriz normal atiende a la cuantificación del número de objetos de cada clase, mientras que la matriz normalizada, la cual nos servirá para analizar los resultados, muestra los datos en forma de porcentaje.

En primer lugar, en ambos modelos se observa que la mayor parte de las detecciones correctas se concentra en la diagonal principal de la matriz, lo que indica que los algoritmos han sido capaces de aprender las características generales de las distintas clases definidas en el conjunto de datos. Esto confirma que el proceso de aprendizaje ha permitido a ambos modelos distinguir, en términos generales, entre las categorías de objetos presentes en las imágenes térmicas. No obstante, el análisis comparativo muestra algunas diferencias relevantes entre ambas arquitecturas. En el caso de YOLOv11, se observa una mayor dispersión de valores fuera de la diagonal principal, lo que indica la presencia de confusiones más frecuentes entre determinadas clases. Estas confusiones se producen principalmente entre las categorías diferenciadas por la distancia, y no por el tamaño. La matriz correspondiente al modelo YOLOv26 indica que el modelo es capaz de reducir la frecuencia de errores de clasificación en comparación con la versión anterior. Cabe destacar que, en contra de lo que se consideraría lógico, ambos modelos coinciden en una mejor detección de objetos a distancia lejana que a corta distancia, con valores que superan el 60%.

Otro aspecto relevante es la presencia de detecciones asociadas a la categoría '*background*', la cual corresponde a predicciones realizadas por el modelo en regiones de la imagen donde no existe un objeto etiquetado, es decir, fondo de la imagen. Estas detecciones representan falsos positivos, y suelen estar relacionadas con patrones térmicos presentes en el entorno marítimo, como reflejos en la superficie del agua o estructuras que generan contrastes térmicos similares a los de los objetos de interés. Asimismo, en algunos casos se identifican falsos negativos, es decir, objetos presentes en la imagen que no han sido detectados por el modelo. Este fenómeno suele producirse cuando los objetos aparecen a gran distancia de la cámara o presentan un contraste térmico reducido respecto al entorno, lo que dificulta su identificación.

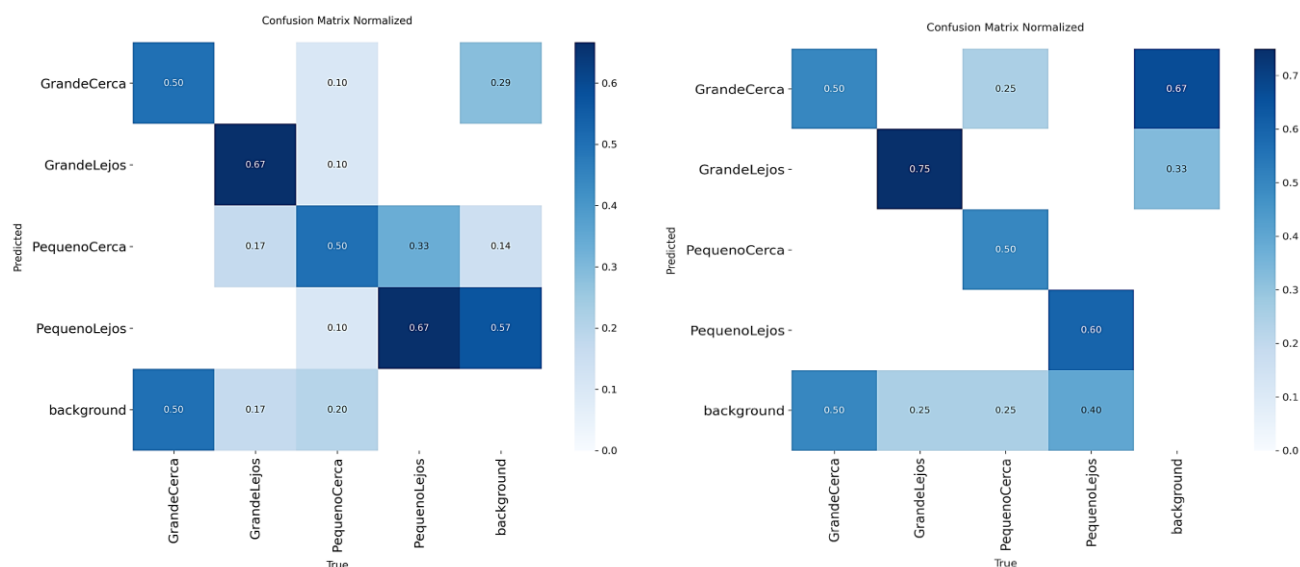


Figura 4.3 Matrices de confusión normalizadas de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) del Escenario 1 (Elaboración propia)

Con el fin de evaluar visualmente el comportamiento del modelo, se generan 16 ejemplos de imágenes correspondientes al conjunto de validación en las que se comparan dos tipos de imágenes: por un lado, las imágenes '*LABEL*' son aquellas que representan las anotaciones originales del conjunto de datos, asignadas durante el proceso de etiquetado; por otro lado, las imágenes '*PRED*', que corresponden a las detecciones generadas automáticamente por el modelo tras el proceso de entrenamiento y validación.

En el caso del modelo YOLOv11 (véase Figura 4.4), se ha observado que el modelo es capaz de detectar correctamente una parte significativa de los objetos presentes en las escenas, generando las correspondientes *bounding boxes* y asignando la clase adecuada en numerosos casos. Sin embargo, también se identifican algunas situaciones en las que el modelo presenta detecciones múltiples sobre un mismo objeto, omisiones de objetos presentes o errores de clasificación entre categorías similares. Estas situaciones suelen aparecer especialmente en objetos de menor tamaño o situados a mayor distancia de la cámara, donde el contraste térmico y la resolución aparente del objeto se reducen. Por su parte, el modelo YOLOv26 (véase Figura 4.5) muestra un comportamiento ligeramente más estable en las predicciones. En las imágenes analizadas se observa una mayor correspondencia entre las etiquetas reales y las predicciones generadas, reduciéndose en algunos casos el número de errores de clasificación y detecciones redundantes. No obstante, ambos modelos presentan ciertas dificultades en situaciones en las que los objetos aparecen a gran distancia o con un tamaño reducido dentro de la imagen, lo que puede provocar que el modelo no genere ninguna detección o que la clase asignada no coincida con la etiqueta real.

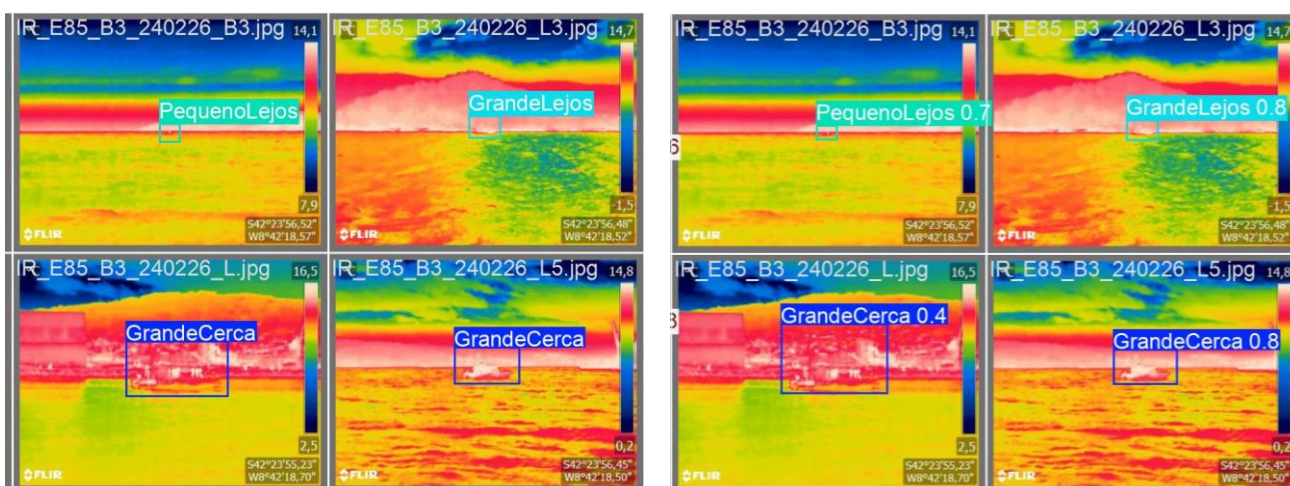


Figura 4.4 Ejemplos de imágenes ‘*LABEL*’ (a la izquierda) y ‘*PRED*’ (a la derecha) del modelo YOLOv11 en el Escenario 1 (Elaboración propia)

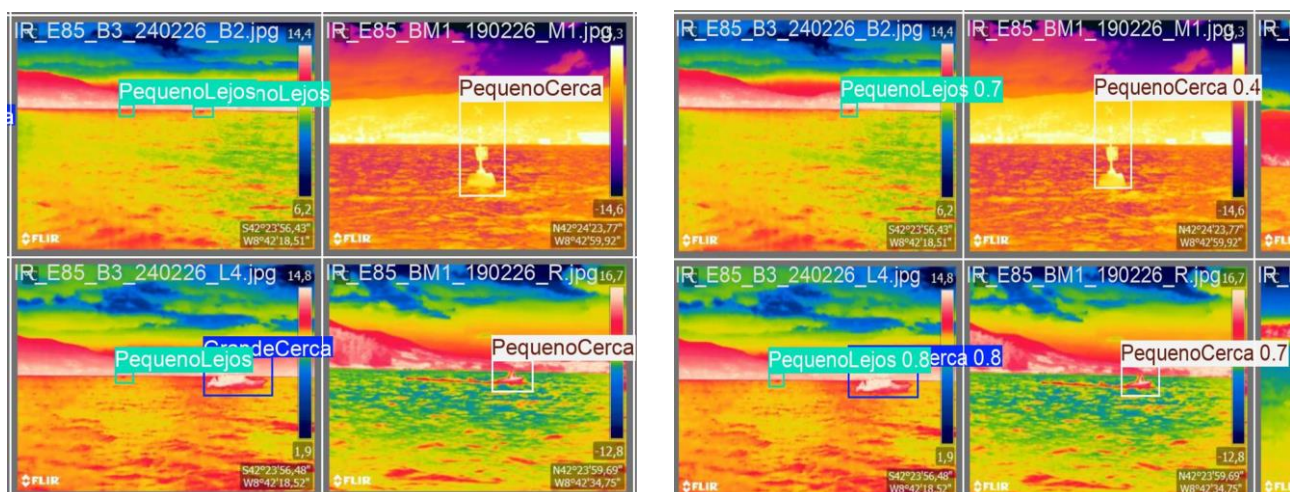


Figura 4.5 Ejemplos de imágenes ‘*LABEL*’ (a la izquierda) y ‘*PRED*’ (a la derecha) del modelo YOLOv26 en el Escenario 1 (Elaboración propia)

Finalmente, el modelo aprendido ha sido aplicado sobre un conjunto de 20 imágenes de test no utilizadas durante el aprendizaje, con el objetivo de analizar el comportamiento de los modelos ante nuevas imágenes y comprobar si las características aprendidas durante el proceso de aprendizaje son capaces de trasladarse correctamente a escenarios no vistos previamente. Este proceso de testeo se ha realizado con el modelo *last.pt* basado en la última iteración, ya que el modelo presenta mejores resultados al haber sido enriquecido con las iteraciones anteriores y presentar una estabilización en las fases finales del entrenamiento. En el caso del modelo YOLOv11, los resultados muestran detecciones correctas especialmente en aquellos que presentan un mayor tamaño. Por su parte, el modelo YOLOv26 presenta un comportamiento similar en términos generales, aunque en varios de los ejemplos analizados se aprecia una mayor estabilidad en las predicciones y una mejor correspondencia entre los objetos presentes en la imagen y las cajas delimitadoras generadas por el modelo.

En la Figura 4.6 se muestran los resultados finales obtenidos en la fase de testeo de ambas versiones del modelo YOLO. En ellos, se incluyen cuatro ejemplos correspondientes a las cuatro clases de objetos establecidas para el estudio de las imágenes.

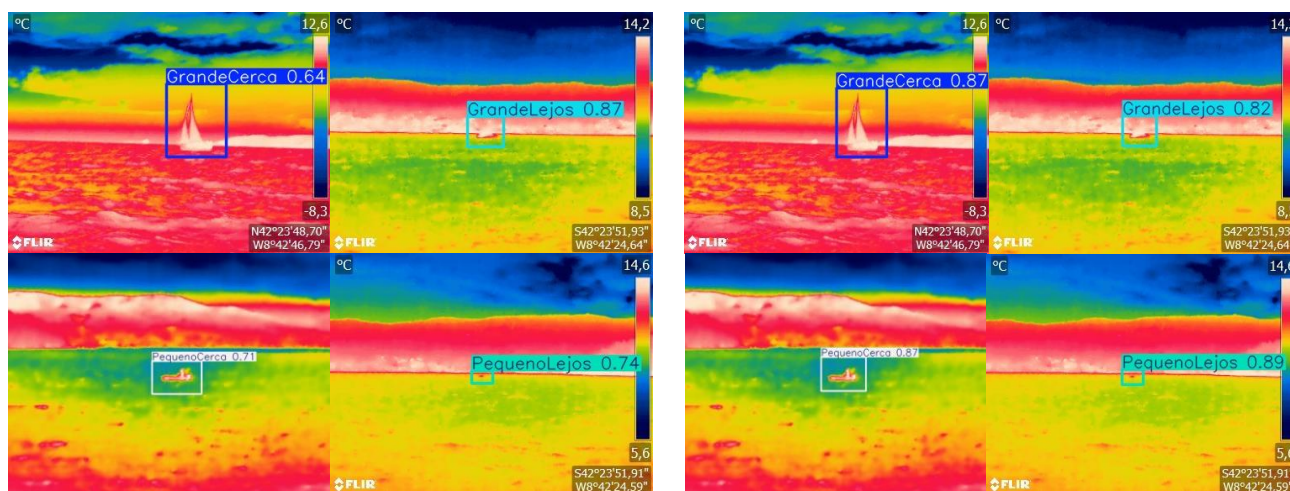


Figura 4.6 Resultados de la fase de testeo del Escenario 1 del modelo YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) (Elaboración propia)

Mask R-CNN: para este modelo, el conjunto de datos se ha dividido de la siguiente forma:

- 100 imágenes para el aprendizaje, las cuales se dividieron manualmente en 80 imágenes de entrenamiento y 20 de validación, pero de forma que las clases de objetos estuvieran distribuidas de forma equitativa para ambos conjuntos.
- 20 imágenes para testeo, las mismas que para el modelo YOLO.

Este modelo, al presentar una mayor complejidad que los modelos YOLO debido a la incorporación de la segmentación de instancias, monitoriza un mayor número de funciones de pérdida y métricas, reflejadas en las siguientes figuras. En primer lugar, se obtienen las pérdidas asociadas a las pérdidas de localización de los *bounding boxes*, a la clasificación y las pérdidas asociadas a la generación de máscaras (Figura 4.7). Al igual que ocurre en el modelo YOLO, las pérdidas de clasificación y localización muestran una tendencia descendente a medida que avanza el entrenamiento, lo que indica que el modelo mejora progresivamente su capacidad para identificar y localizar los objetos presentes en el conjunto de datos. La evolución de la pérdida de máscara permite comprobar cómo el modelo aprende gradualmente a delimitar con mayor exactitud las regiones correspondientes a cada objeto.



Figura 4.7 Gráficas de pérdidas del modelo Mask R-CNN durante la fase de entrenamiento en el Escenario 1 (Elaboración propia)

Durante el entrenamiento también se monitorizan las pérdidas asociadas a la etapa de la *Region Proposal Network (RPN)*, como puede observarse en la Figura 4.8, que reflejan la capacidad del modelo para proponer regiones que realmente contienen objetos relevantes dentro de la imagen. Una reducción progresiva de estas pérdidas indica que el modelo mejora su capacidad para identificar correctamente las posibles regiones de interés. En conjunto, el análisis de las distintas curvas de pérdida muestra una convergencia progresiva del entrenamiento, lo que sugiere que el modelo ha sido capaz de ajustar sus parámetros de forma estable a partir del conjunto de datos disponible.

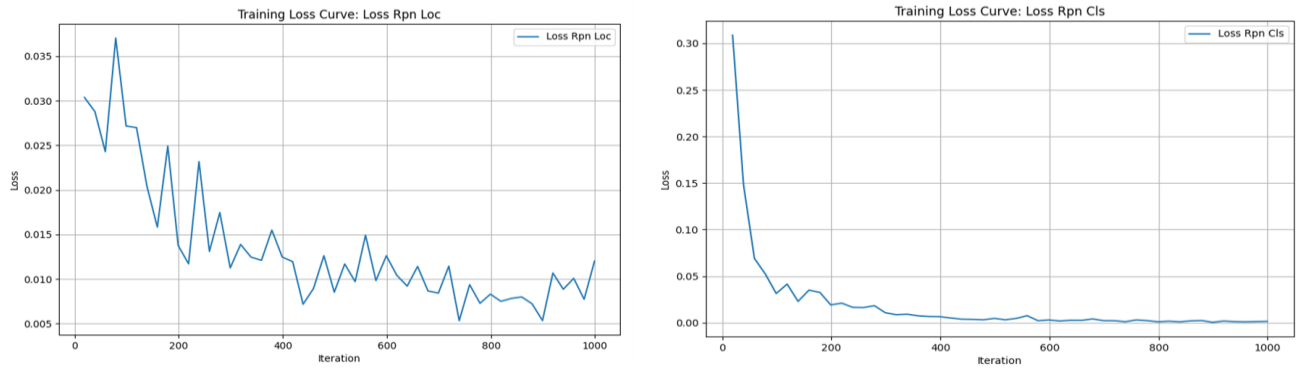


Figura 4.8 Gráficas de pérdidas asociadas a la RPN en el Escenario 1 (Elaboración propia)

De forma similar al modelo YOLO, las gráficas que representan métricas del conjunto de validación muestran pequeñas oscilaciones entre iteraciones, especialmente en las primeras fases de aprendizaje, lo cual sigue siendo habitual en estos modelos. En Mask R-CNN, estas métricas aparecen divididas en dos grupos principales: *bbox* (véase Figura 4.9) y *segm* (véase Figura 4.10), ya que realiza dos tareas diferentes en el conjunto de validación. La primera de ellas evalúa la calidad de la detección del objeto, mientras que la referente a la segmentación de instancias analizan la precisión de las máscaras de segmentación generadas para cada objeto. En este caso, los valores obtenidos son menores que los del YOLO, debido a la mayor complejidad de las tareas que realiza este modelo. Tras el aprendizaje, se ronda en un 35%, 70% y 30% en la detección, y en un 35%, 65% y 25% en la segmentación, según las métricas *AP*, *AP50* y *AP75*, respectivamente.

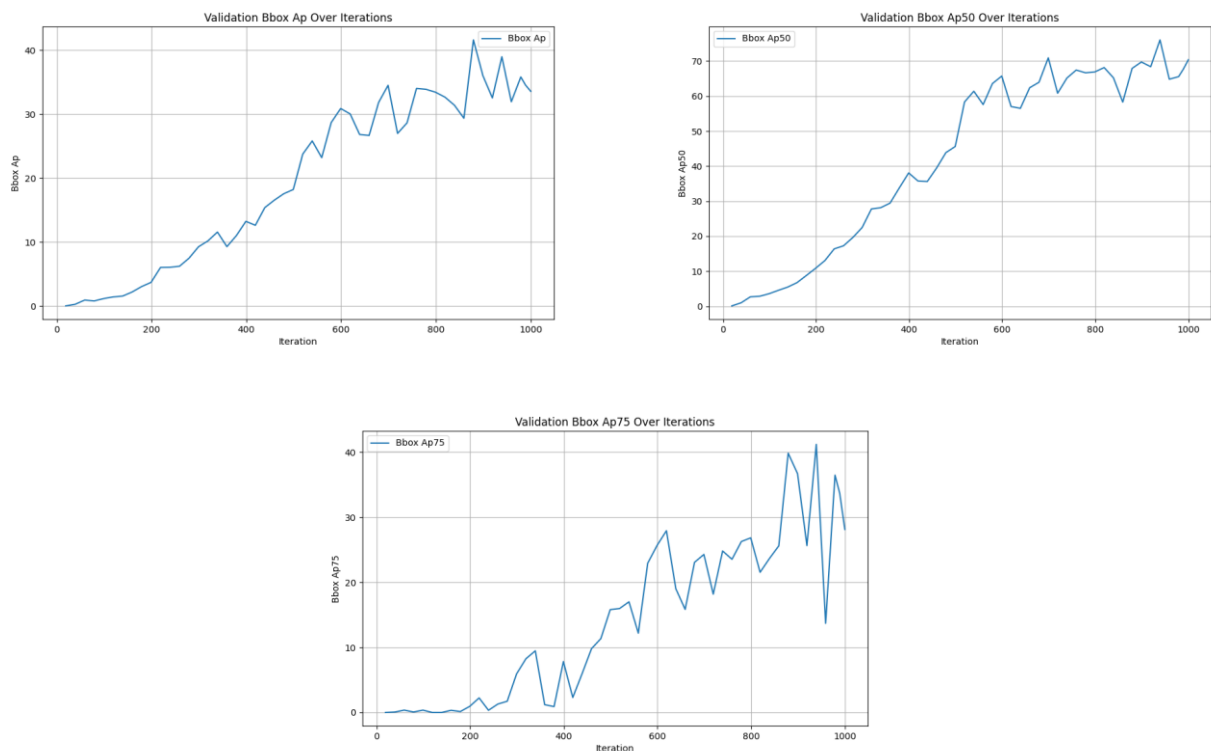


Figura 4.9 Métricas de evaluación del bounding box del conjunto de validación del Escenario 1 (Elaboración propia)

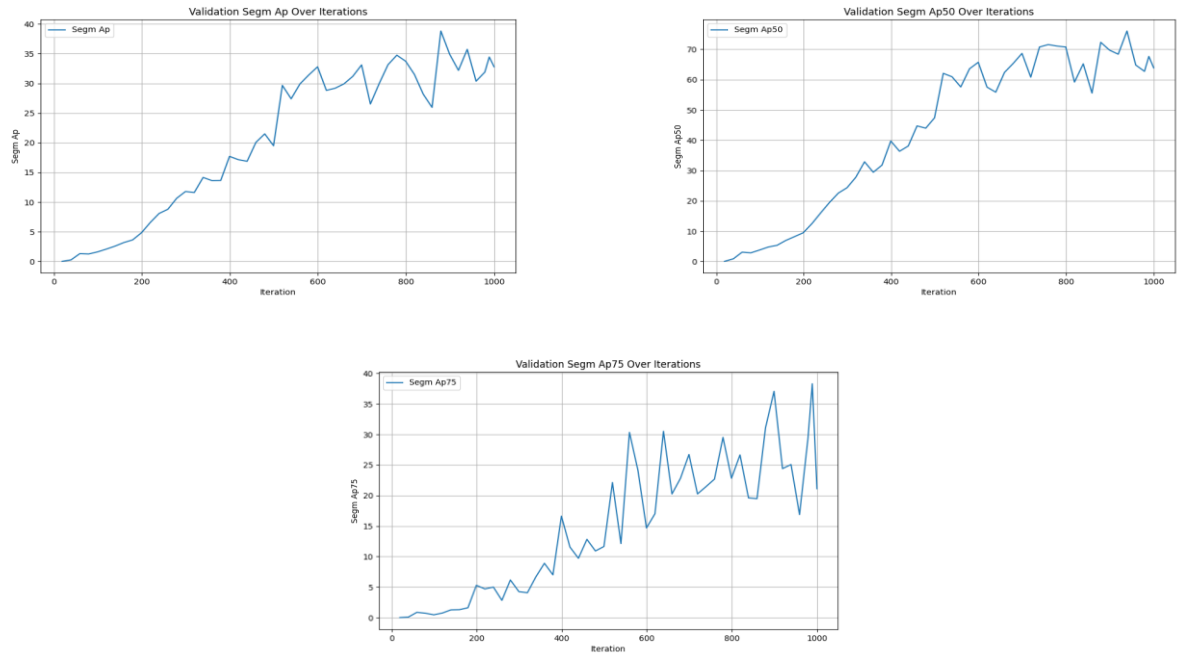


Figura 4.10 Métricas de evaluación de la segmentación del conjunto de validación del Escenario 1 (Elaboración propia)

Tal y como las genera YOLO, el modelo Mask R-CNN muestra las matrices de confusión normal y normalizada. En comparación con los modelos YOLO analizados previamente, la matriz de confusión de Mask R-CNN presenta un comportamiento ligeramente diferente debido a la propia naturaleza del modelo. Al incorporar un proceso adicional de segmentación de instancias, el modelo debe aprender no solo a detectar la presencia del objeto, sino también a delimitar con precisión su contorno. Este requisito adicional aumenta la complejidad del problema y puede explicar la presencia de ciertos errores de clasificación o detección observados en la matriz.

Tal y como se observa en la Figura 4.11, este modelo presenta un mayor número de errores definidos como *background* debido principalmente al uso de la *RPN*, que hace que el modelo evalúe muchas más regiones de las que finalmente se consideran detecciones válidas. Sin embargo, cabe destacar la capacidad de detección de objetos de pequeño tamaño a larga distancia (60%), rondando el resto de las clases entre el 25% y el 43%.

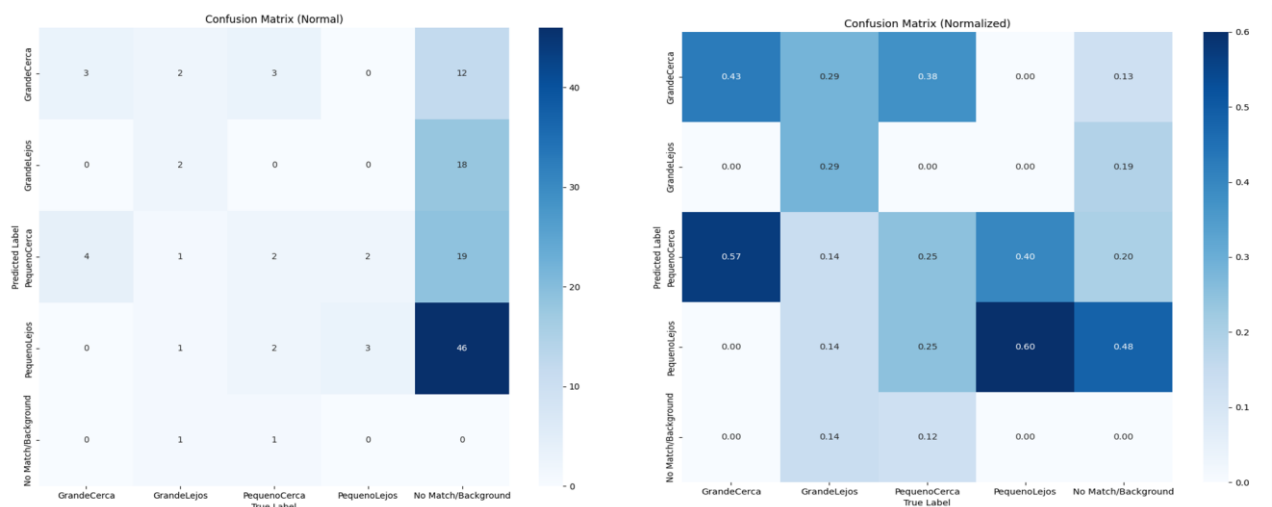


Figura 4.11 Matriz de confusión normal (a la izquierda) y normalizada (a la derecha) del modelo Mask R-CNN en el Escenario 1 (Elaboración propia)

En cuanto a la comparación entre las etiquetas reales y las predicciones del modelo en el conjunto de validación (véase Figura 4.12), el análisis visual de los ejemplos muestra que el modelo es capaz de identificar correctamente varios de los objetos presentes en las escenas, generando máscaras que se ajustan razonablemente bien al contorno de los objetos detectados. En particular, los objetos de mayor tamaño o aquellos que presentan un contraste térmico más marcado respecto al fondo son detectados con mayor facilidad por el modelo. A pesar de ello, presenta errores notables en la clasificación de los objetos, aunque la delimitación de estos resulta positiva, hecho que coincide con el análisis previo realizado en las matrices de confusión.

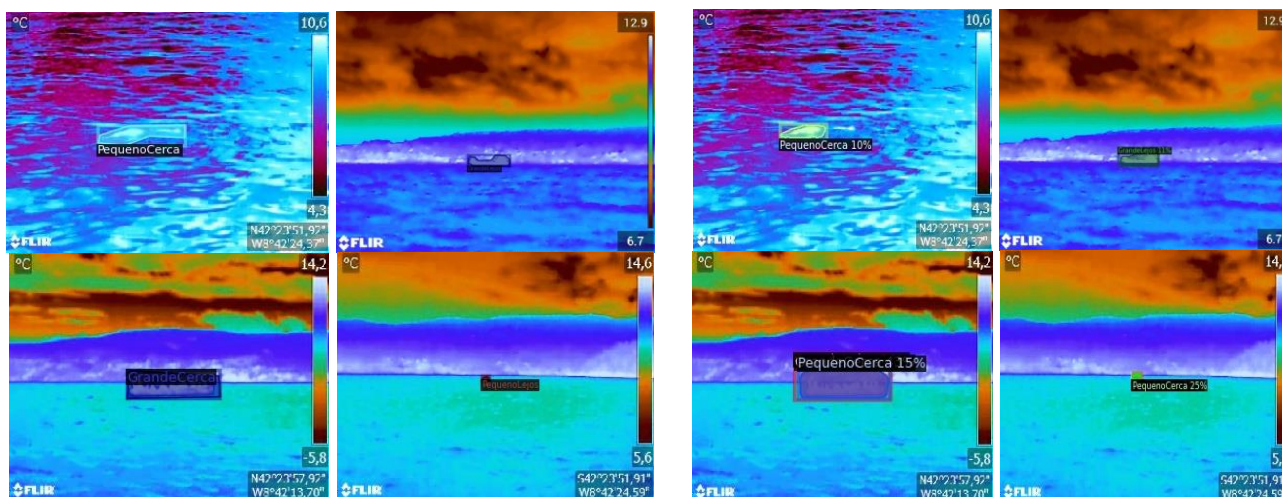


Figura 4.12 Ejemplos de imágenes ‘LABEL’ (a la izquierda) y ‘PRED’ (a la derecha) del modelo Mask R-CNN en el Escenario 1 (Elaboración propia)

A pesar de los resultados variados en la comparación previa, el proceso de testeo resulta de forma muy favorable, generando segmentaciones más precisas del contorno del objeto, como puede observarse en la Figura 4.13. No obstante, el modelo presenta mayores dificultades cuando se trata de discernir entre el tamaño de los objetos, en este caso el resultado superior izquierdo se debería corresponder con una clase de objeto ‘PequenoCerca’ aunque la detección y segmentación haya sido correcta.

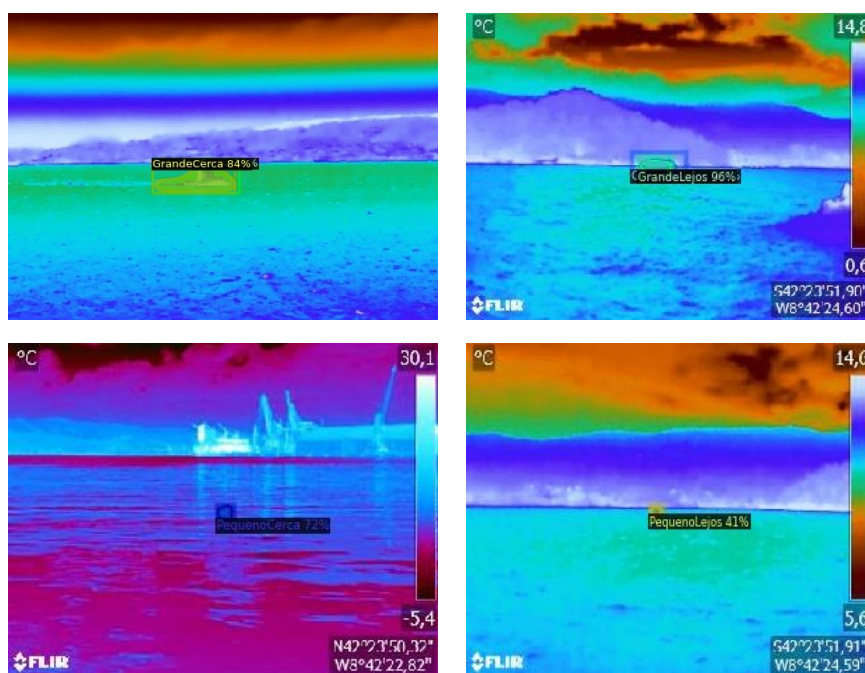


Figura 4.13 Resultados de la fase de testeo del modelo Mask R-CNN en el Escenario 1 (Elaboración propia)

4.2 Resultados en el Escenario 2

En este segundo escenario se presentan los resultados obtenidos sobre un nuevo conjunto de datos que incluye imágenes procedentes de la FLIR E-85 y la FLIR One Edge. A diferencia del escenario anterior, en este caso se incorporan las imágenes obtenidas con un dispositivo de menor coste y menor resolución térmica. La inclusión de imágenes de ambas cámaras permite evaluar el comportamiento de los modelos ante diferentes niveles de calidad en la información térmica, así como analizar la capacidad de los algoritmos para generalizar cuando se enfrentan a datos procedentes de sensores con características distintas.

YOLO: para este caso, el conjunto de datos se ha dividido de la siguiente forma:

- 100 imágenes para el aprendizaje de las cuales, 60 pertenecen a la FLIR E-85 y 40 a la FLIR One Edge. Al igual que para el escenario 1, 80 (80%) son para entrenamiento y 20 (20%) para validación, asignadas de forma aleatoria y garantizando que en cada conjunto existiera equilibrio entre el número de imágenes de una cámara y el número de imágenes de la otra.
- 20 imágenes para testeo no utilizadas previamente en el aprendizaje, 10 correspondientes a cada cámara.

El análisis de las curvas de pérdida (véase Figura 4.14) muestra una evolución similar a la observada en el escenario anterior con respecto al conjunto de entrenamiento. Sin embargo, en comparación con el escenario anterior se observan oscilaciones en las curvas de validación, lo que puede atribuirse a la mayor heterogeneidad del conjunto de datos al incluir imágenes procedentes de dos sensores distintos. Las diferencias en resolución térmica y calidad de imagen introducen mayor variabilidad en las características visuales de los objetos, lo que puede dificultar el proceso de aprendizaje del modelo, sobre todo si los datos de entrada son de peor calidad tal y como son las imágenes de la FLIR One Edge. El modelo YOLOv26 presenta una evolución más estable de las métricas de validación, lo que indica una mayor robustez frente a la variabilidad introducida por las imágenes de diferentes sensores. En cuanto a las métricas, como se observa en la Figura 4.15, indican que el modelo mantiene una capacidad adecuada para detectar los objetos definidos en el estudio, aunque se aprecia una ligera reducción del rendimiento respecto al escenario anterior, especialmente en imágenes procedentes de la cámara de menor resolución. Se pasa de una ‘*Precision*’ del 80% del Escenario 1 a cerca del 60% en el YOLOv11, aunque se mantiene el valor en el YOLOv26, en el Escenario 2. Referido al ‘*Recall*’, la capacidad de cobertura en la detección de los objetos existentes en cada imagen queda en valores similares en ambas versiones con respecto al Escenario 1, es decir, en torno al 60%. Por tanto, la bajada de la métrica *mAP50* del 70% del Escenario 1 hacia el 60% (bajada que solo afecta al YOLOv11) del Escenario 2 se debe principalmente al aumento del número de falsos positivos (‘*Precision*’) en vez de falsos negativos ‘*Recall*’, es decir, la cobertura abarcada en la detección de todos los objetos se mantiene, pero se aumenta en el número de detecciones fallidas. De ahí también la bajada del 40% al 30% en cuanto a la métrica *mAP50-95* en el YOLOv11 aunque, de nuevo, el YOLOv26 muestra una gran robustez en su aprendizaje a pesar de la introducción de un porcentaje de datos categorizados como mala calidad.

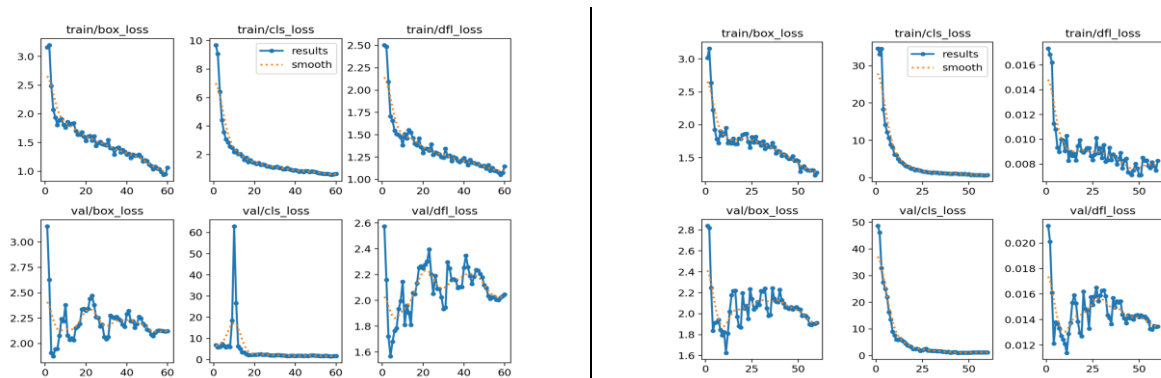


Figura 4.14 Resultados gráficos de las funciones de pérdida (*loss*) de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) en el Escenario 2 (Elaboración propia)

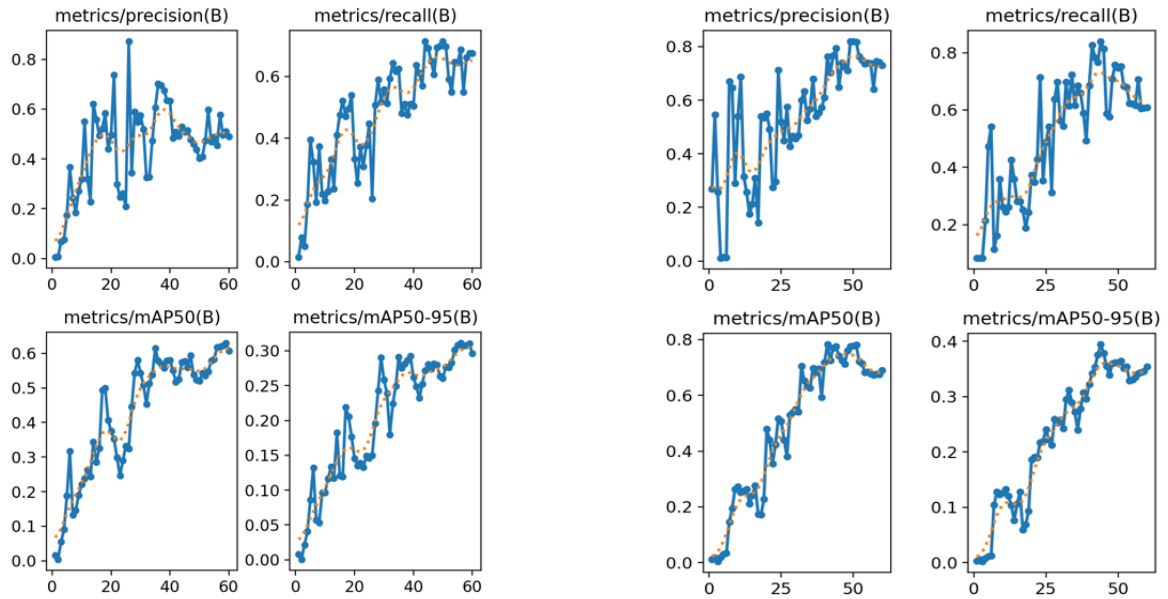


Figura 4.15 Resultado de las métricas de evaluación de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) en el Escenario 2 (Elaboración propia)

Los resultados de las matrices de confusión (véase Figura 4.16) muestran que el modelo continúa clasificando correctamente una parte significativa de los objetos. En este caso, se observa como la detección en el modelo YOLOv26 alcanza altos valores no logrados en el anterior escenario, pasando del 50%, 75%, 50% y 60% en ‘GrandeCerca’, ‘GrandeLejos’, ‘PequenoCerca’ y ‘PequenoLejos’ en el Escenario 1; al 100%, 100%, 87% y 50% en ‘GrandeCerca’, ‘GrandeLejos’, ‘PequenoCerca’ y ‘PequenoLejos’ en el Escenario 2. Además, presenta una clara diferencia en cuanto a la variabilidad de detecciones con respecto a la versión 11, con incrementos hasta del 50% en ‘GrandeLejos’, además de un menor número de falsas detecciones. No obstante, se observan algunas confusiones entre clases, especialmente en aquellos casos en los que los objetos aparecen con menor definición térmica. Este comportamiento es más frecuente en imágenes capturadas con la FLIR One Edge, donde la resolución y el contraste térmico son inferiores, perjudicando sobre todo en la detección de objetos clasificados como ‘PequenoLejos’. Por tanto, este Escenario 2 presenta mejor detectabilidad a distancias cortas a diferencia del Escenario 1, y la indiferencia del tamaño en el Escenario 1, sí que es ligeramente sensible en este Escenario 2, teniendo mejores resultados en objetos de mayor tamaño (los catalogados como ‘Grande’).



Figura 4.16 Matrices de confusión normalizadas de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) del Escenario 2 (Elaboración propia)

En cuanto a la comparación de las etiquetas reales y las predichas, se identifican algunos casos en los que el modelo presenta detecciones incompletas o clasificaciones incorrectas, especialmente cuando los objetos aparecen a mayor distancia o con un contraste térmico reducido, como se observa en las imágenes seleccionadas de la FLIR One Edge en la Figura 4.17 y la Figura 4.18.

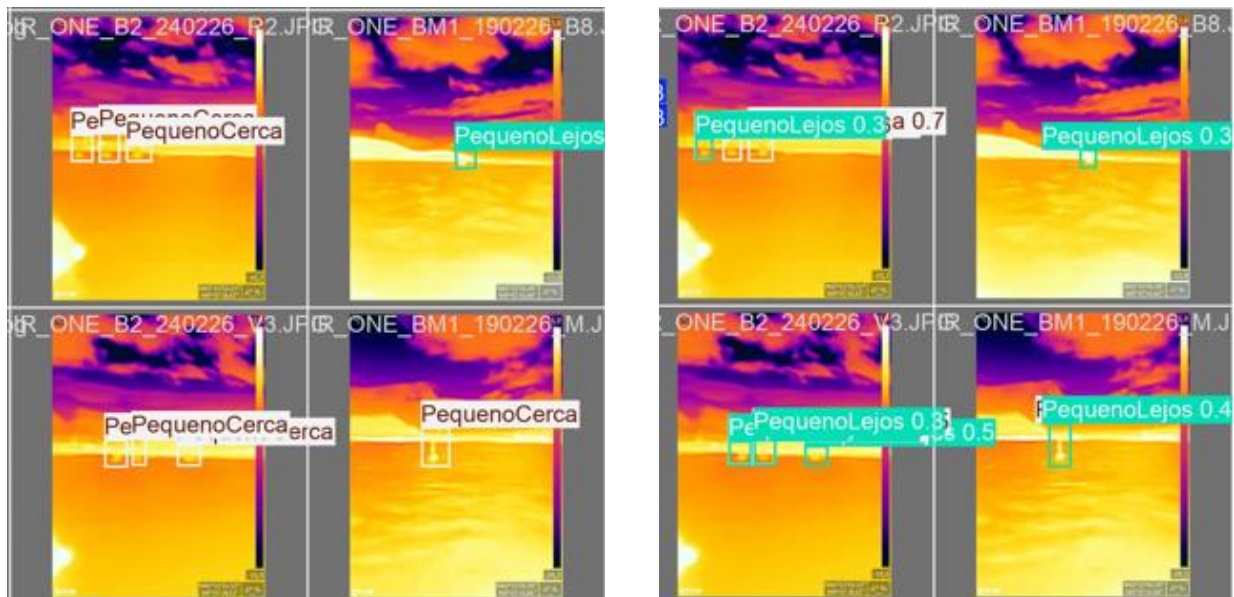


Figura 4.17 Ejemplos de imágenes ‘*LABEL*’ (a la izquierda) y ‘*PRED*’ (a la derecha) del modelo YOLOv11 en el Escenario 2 (Elaboración propia)

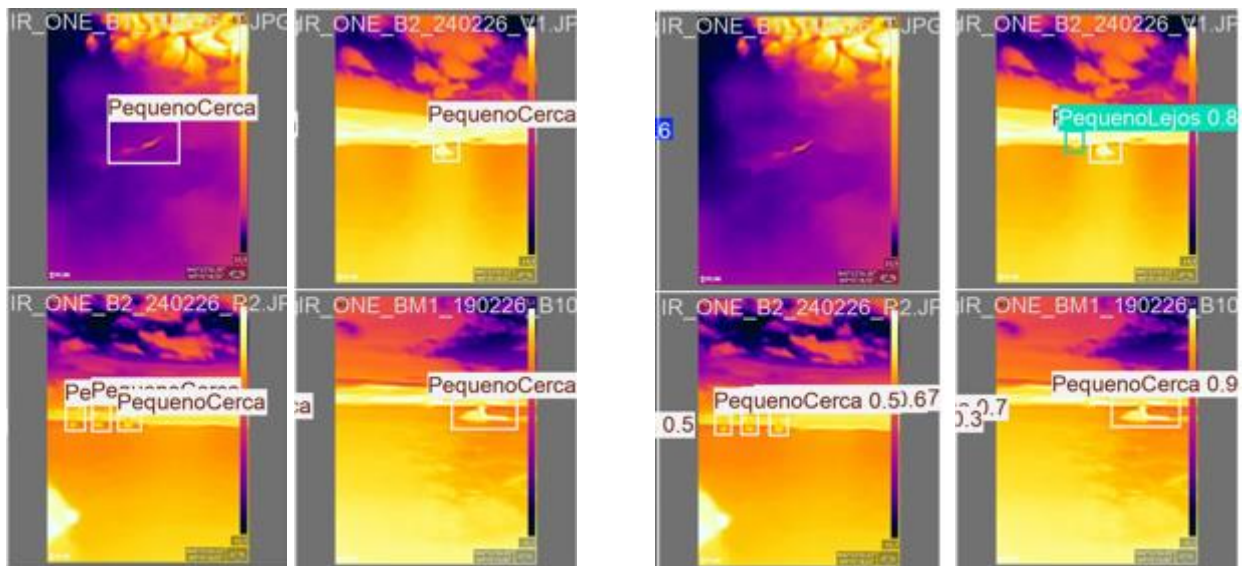


Figura 4.18 Ejemplos de imágenes ‘*LABEL*’ (a la izquierda) y ‘*PRED*’ (a la derecha) del modelo YOLOv26 en el Escenario 2 (Elaboración propia)

Las imágenes sometidas al proceso de testeo demuestran una mayor capacidad de generalización detectando correctamente varios de los objetos presentes, incluso en imágenes capturadas con la cámara de menor resolución, sobre todo con el YOLOv26. Esto sugiere que las mejoras introducidas en la arquitectura de YOLOv26 contribuyen a una mayor robustez frente a variaciones en la calidad de los datos, mientras que el modelo YOLOv11 presenta mayores dificultades para esta tarea.

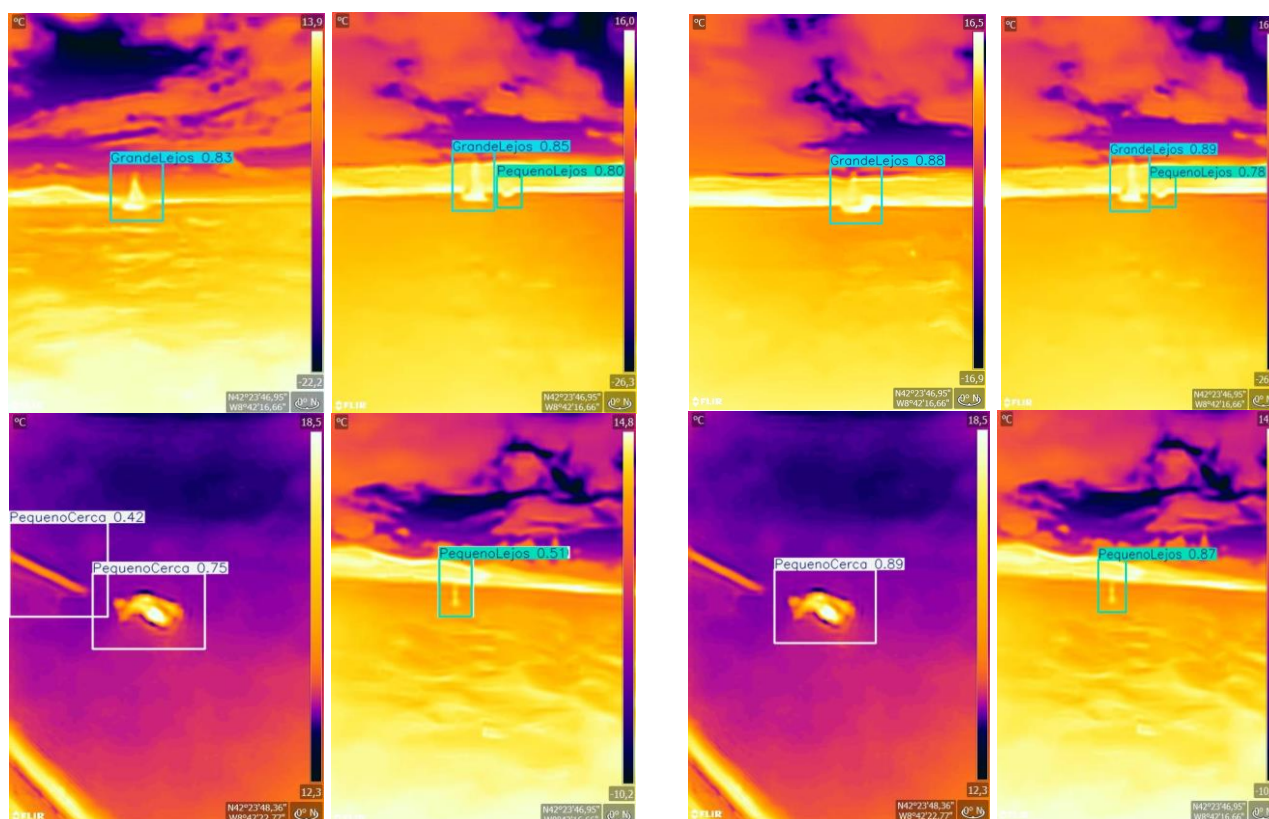


Figura 4.19 Resultados de la fase de testeo del Escenario 2 del modelo YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) (Elaboración propia)

Mask R-CNN: para este caso de estudio, el conjunto de datos quedó distribuido de la siguiente forma:

- 100 imágenes para el aprendizaje, las cuales se dividieron manualmente en 80 imágenes de entrenamiento y 20 de validación. Entre las 80 de entrenamiento se incluyen 50 imágenes de la FLIR E-85 y 30 de la FLIR One Edge, mientras que el conjunto de validación cuenta con 10 imágenes de cada cámara, al igual que en la prueba del modelo YOLO.
- 20 imágenes para testeo utilizadas de la misma forma para el modelo YOLO previo.

En primer lugar, sigue apareciendo una tendencia descendente en las funciones de pérdida, siendo capaz el modelo de aprender las características relevantes del conjunto de datos. Atendiendo a las curvas de la Figura 4.20 Gráficas de pérdidas del modelo Mask R-CNN durante la fase de entrenamiento en el Escenario 2 (Elaboración propia) Figura 4.20 y la Figura 4.21, en este escenario se observa una notable similitud en las pérdidas del entrenamiento con respecto al Escenario 1. A pesar de la dificultad añadida de aprender patrones consistentes a partir de datos más heterogéneos, los datos resultantes muestran pérdidas cercanas al 0 a medida que se llega al final del entrenamiento.



Figura 4.20 Gráficas de pérdidas del modelo Mask R-CNN durante la fase de entrenamiento en el Escenario 2 (Elaboración propia)

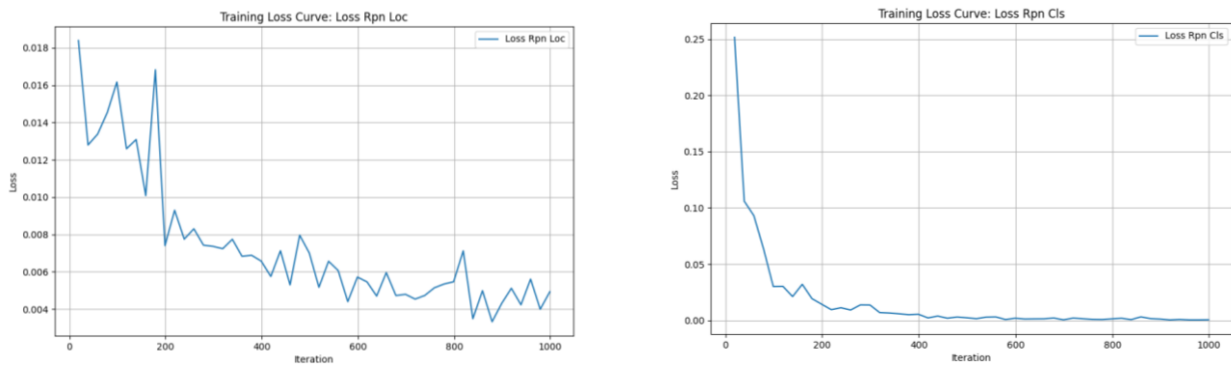


Figura 4.21 Gráficas de pérdidas asociadas a la *RPN* en el Escenario 2 (Elaboración propia)

Según las métricas de validación reflejadas en la Figura 4.22 y la Figura 4.23, el rendimiento en el Escenario 2 es similar al Escenario 1. Se mantiene la tendencia del 35%, 70% y 30% en la detección como en el Escenario 1, y se mantiene también el 35%, 65% y 25% en la segmentación, según las métricas *AP*, *AP50* y *AP75*, respectivamente.

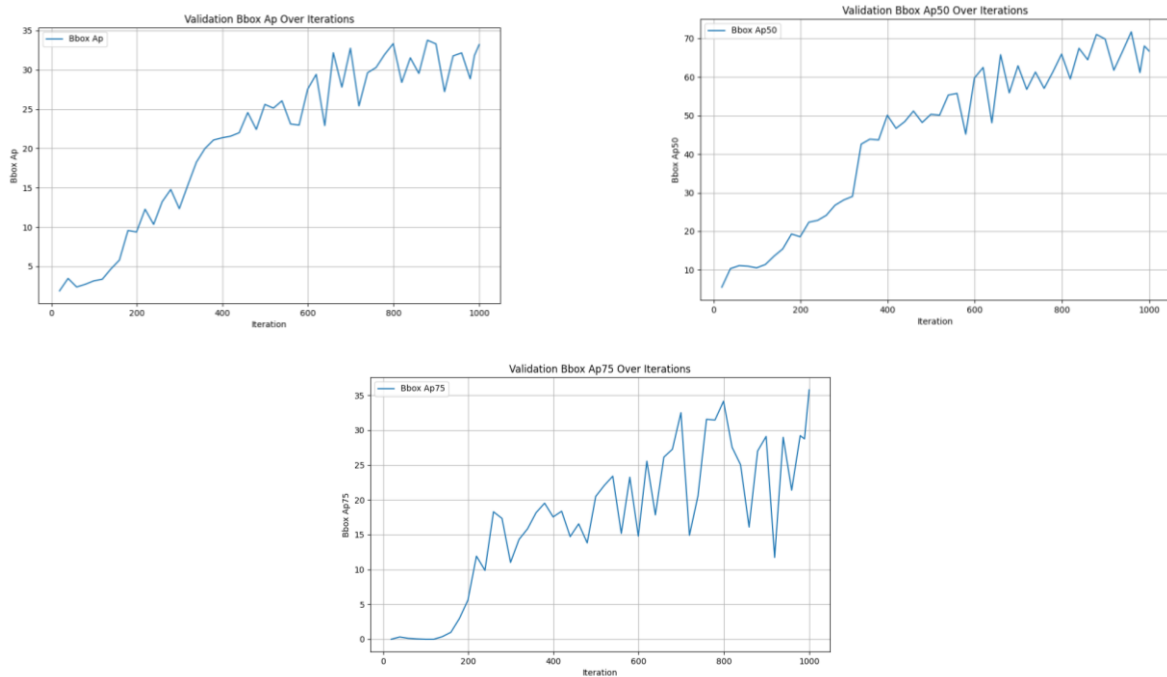


Figura 4.22 Métricas de evaluación del *bouding box* del conjunto de validación del Escenario 2 (Elaboración propia)

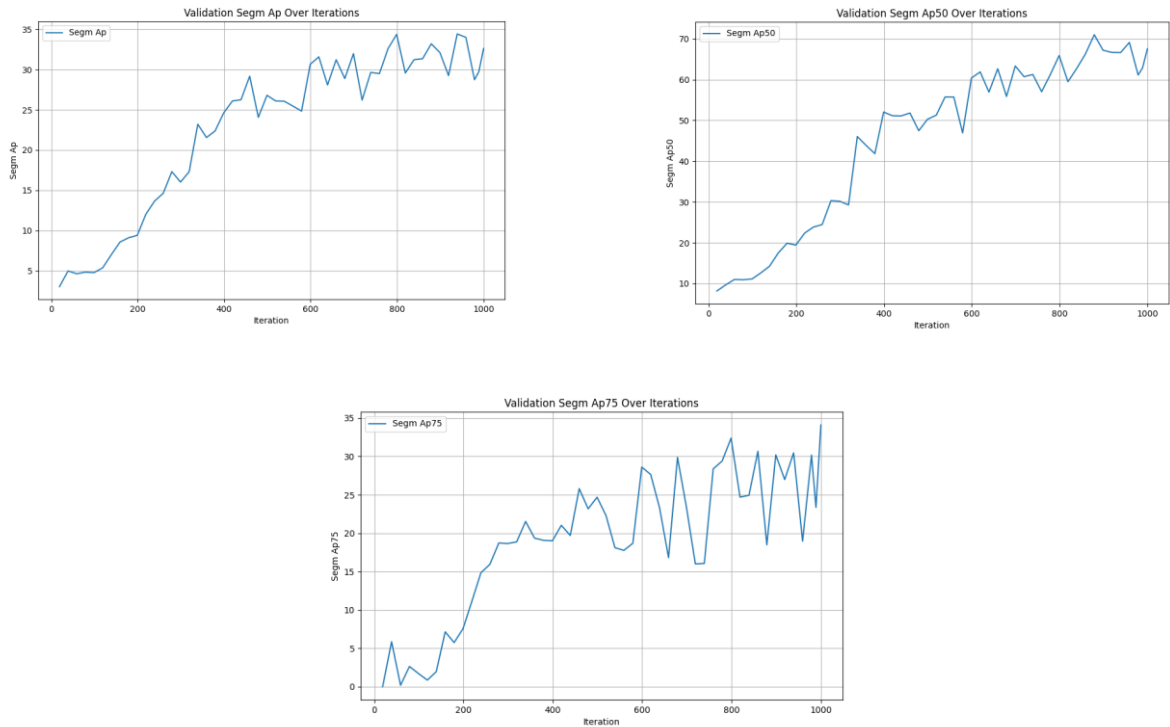


Figura 4.23 Métricas de evaluación de la segmentación del conjunto de validación del Escenario 2 (Elaboración propia)

Lo que se puede apreciar en la Figura 4.24 es un incremento en las predicciones asociadas a la categoría *background* con respecto al Escenario 1. A diferencia del Escenario 1, los mayores valores de detecciones son obtenidos en la clase *Cerca*, pasando del 43, 29%, 25% y 60% en ‘*GrandeCerca*’, ‘*GrandeLejos*’, ‘*PequenoCerca*’ y ‘*PequenoLejos*’ en el Escenario 1; al 60%, 30%, 78% y 33% en ‘*GrandeCerca*’, ‘*GrandeLejos*’, ‘*PequenoCerca*’ y ‘*PequenoLejos*’ en el Escenario 2.

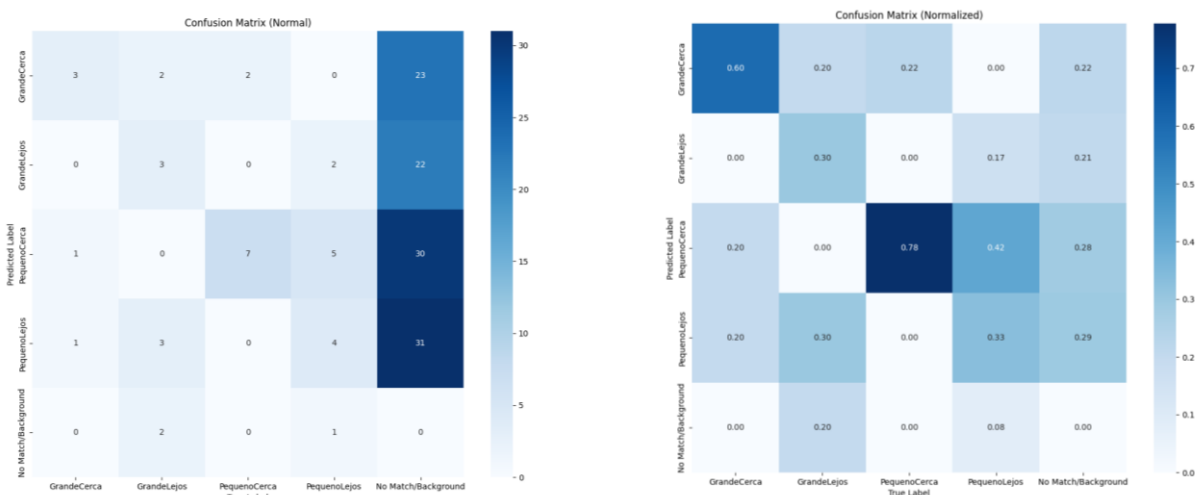


Figura 4.24 Matriz de confusión normal (a la izquierda) y normalizada (a la derecha) del modelo Mask R-CNN en el Escenario 2 (Elaboración propia)

Los resultados visuales de las imágenes ‘*LABEL*’ y ‘*PRED*’ ponen de manifiesto algunos de los errores reflejados en las matrices de confusiones, como la confusión entre el tamaño real de los objetos detectados. En algunos casos se muestran superposiciones de las máscaras, que incluyen más de una clase de objeto, pero la delimitación de los contornos sigue realizándose de forma precisa, como puede observarse en la Figura 4.25.

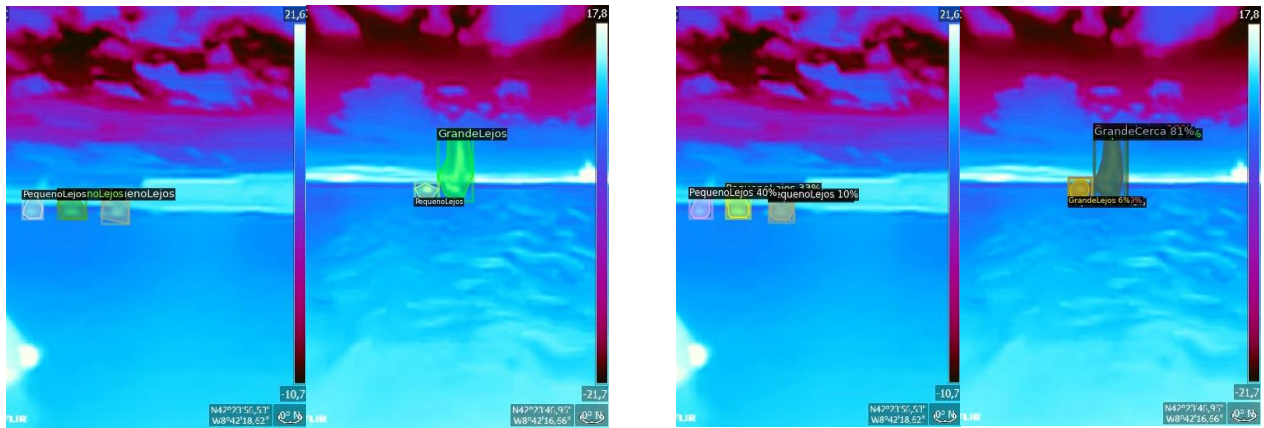


Figura 4.25 Ejemplos de imágenes ‘*LABEL*’ (a la izquierda) y ‘*PRED*’ (a la derecha) del modelo Mask R-CNN en el Escenario 2 (Elaboración propia)

Tras el proceso final de testeo sobre las imágenes no utilizadas, se comprueba que el modelo trabaja de forma coherente. En el Escenario 2 se observa una ligera degradación del rendimiento, especialmente en la detección de objetos pequeños o lejanos. No obstante, el modelo sigue siendo capaz de identificar correctamente los objetos más destacados en la escena, lo que indica que mantiene una capacidad razonable de generalización a pesar de la mayor complejidad del conjunto de datos, como se observa en la Figura 4.26, recordando que dicha figura representa ejemplos de detección de imágenes adquiridas con la FLIR One Edge, las mismas que las utilizadas en el testeo del YOLO.

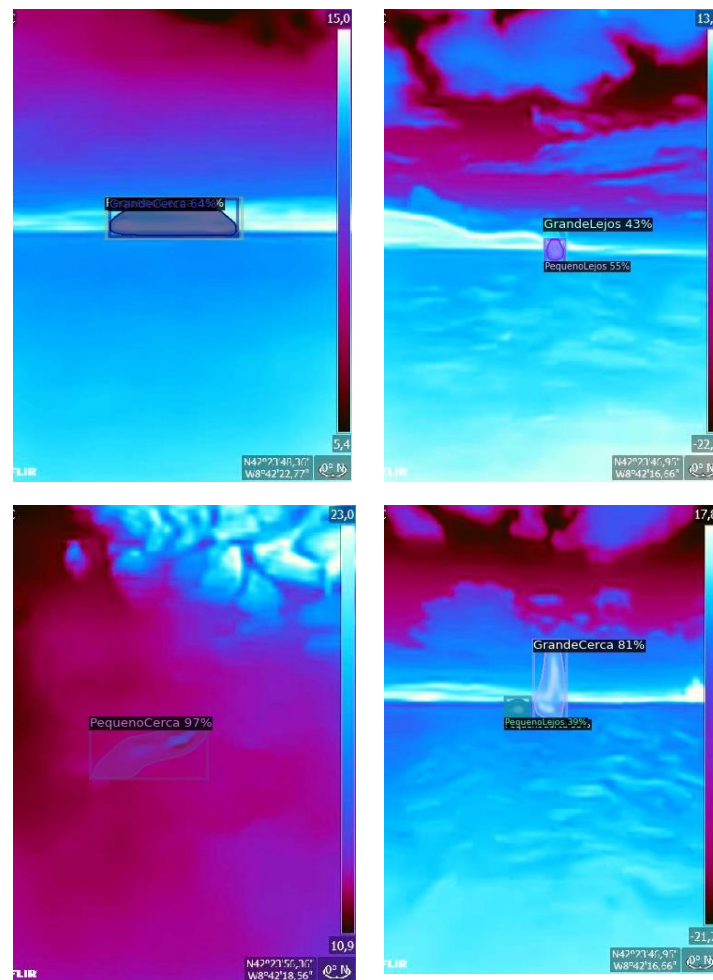


Figura 4.26 Resultados de la fase de testeo del modelo Mask R-CNN en el Escenario 2 (Elaboración propia)

4.3 Resultados en el Escenario 3

En este tercer y último escenario se presentan los resultados obtenidos sobre un nuevo conjunto de datos que incluye imágenes procedentes de la FLIR E-85, de la FLIR One Edge e imágenes de espectro visible. A diferencia de los escenarios anteriores, en este caso se incorporan las imágenes visibles con el objetivo de aportar información adicional relacionada con texturas, formas y detalles estructurales, que no siempre están presentes en el espectro infrarrojo. Sin embargo, también introduce una mayor complejidad en el proceso de aprendizaje, ya que el modelo debe adaptarse a dos dominios de imagen diferentes.

YOLO: para este caso, el conjunto de datos se ha dividido de la siguiente forma:

- 100 imágenes para el aprendizaje: 40 pertenecen a la FLIR E-85, 30 a la FLIR One Edge y 30 imágenes de espectro visible. De igual modo que para los primeros escenarios, 80 (80%) son para entrenamiento y 20 (20%) para validación, asignadas de forma aleatoria, pero asegurando un reparto proporcional tanto de cada tipo de imagen como de cada tipo de clase de objeto en cada conjunto.
- 20 imágenes para testeo no utilizadas previamente en el aprendizaje, de las cuales 5 son de la FLIR E-85, 5 de la FLIR One Edge y 10 de espectro visible.

Los resultados de las gráficas de pérdidas en el conjunto de entrenamiento (véase Figura 4.27) muestran un ligero empeoramiento del rendimiento en este escenario en el caso del YOLOv11, ya que no se obtiene un valor cercano al 0 como en escenarios anteriores, posiblemente debido a la combinación de imágenes de diferente espectro, induciendo diferencias significativas en las características visuales de los objetos. El modelo YOLOv26 sigue mostrando, en general, un comportamiento robusto, y ambos modelos con buenas prestaciones según las pérdidas referidas al conjunto de validación. En conjunto (véase Figura 4.28), el modelo mantiene valores aceptables de *precisión*, *recall* y *mAP*, sobre todo en el YOLOv11, a diferencia de los escenarios previos. En el YOLOv26, como en el Escenario 2 para el YOLOv11, se baja la '*Precision*' hasta el 60%, pero en el '*Recall*' se sube hasta un 70%, siendo la tendencia hasta ahora de un 60%. En el YOLOv11 incluso ese dato se aprecia mejor, manteniendo la '*Precision*' en torno al 80% como en los dos escenarios previos, y el '*Recall*' subiendo hasta prácticamente el 80%. Con ello, se ve que no hay un claro ganador entre los dos modelos de YOLO empleados, donde ambos ofrecen buenos rendimientos, sobre todo en este Escenario 3 al enriquecer los modelos con el aumento de distintas tipologías de datos de entrada (imagen térmica de alta/baja calidad e imagen visible). Tanto es así que el *mAP50* supera el 70% para el YOLOv26 y alcanza el 80% para el YOLOv11, donde el techo hasta ahora era del 70%. Lo mismo ocurre con *mAP50-95*, siguiendo prácticamente con el 40% en el YOLOv26, y superando claramente ese techo del 40% con el YOLOv11, techo no superado hasta este momento.

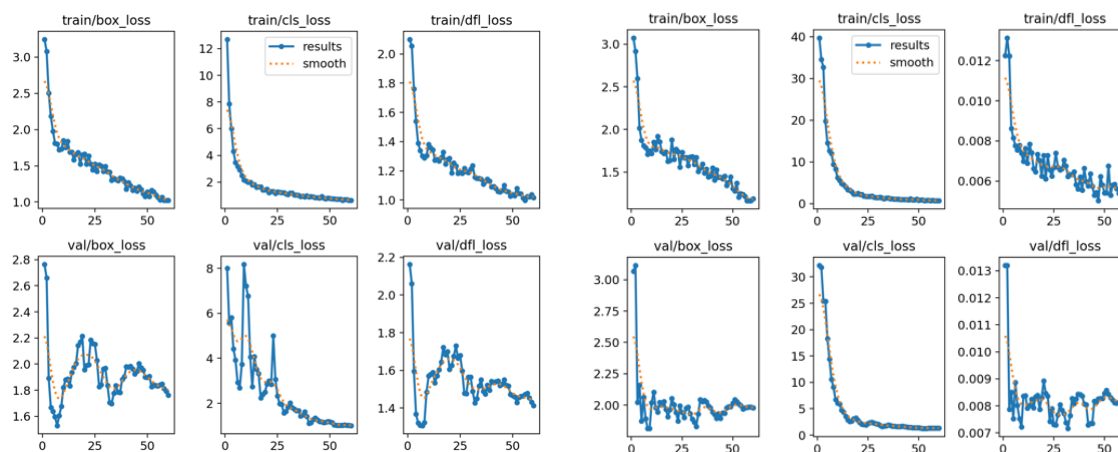


Figura 4.27 Resultados gráficos de las funciones de pérdida (*loss*) de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) en el Escenario 3 (Elaboración propia)

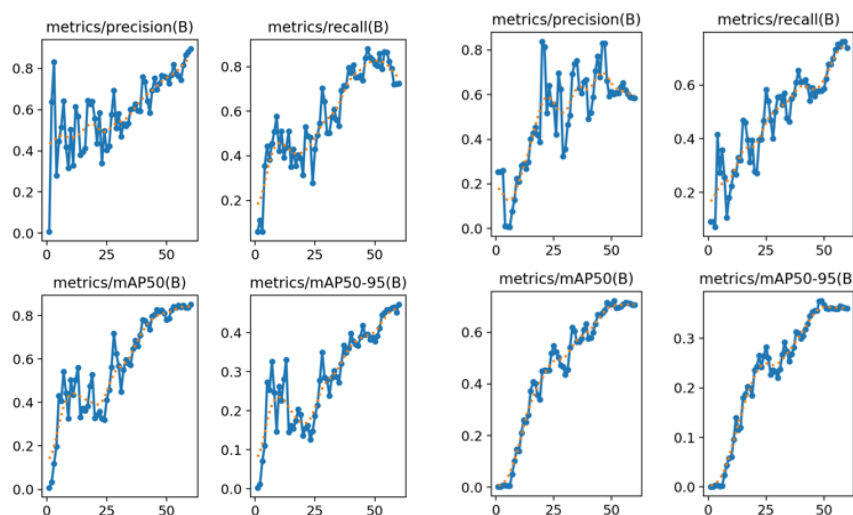


Figura 4.28 Resultado de las métricas de evaluación de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) en el Escenario 3 (Elaboración propia)

Estos mejores rendimientos mencionados se reflejan también en la Figura 4.29, en donde la matriz de confusión de YOLOv11 presenta una elevada capacidad de detección coincidente con la clase real del objeto llegando a valores del 89% en el caso de los objetos ‘GrandeLejos’, seguido del 87%, 80% y 67% para ‘PequenoCerca’, ‘GrandeCerca’ y ‘PequenoLejos’, respectivamente, mientras que el YOLOv26 alcanza un 92% en los denominados ‘PequenoCerca’, seguido del 56% en ‘GrandeLejos’ y ‘PequenoLejos’ y fallando totalmente en ‘GrandeCerca’ considerando un 60% como otro tipo de clase y un 40% como fondo. De ahí se realza que, a diferencia del Escenario 1 y 2, el modelo más robusto es el YOLOv11, y con mayor diferencia con respecto a las dos comparaciones anteriores. Además, el YOLOv11 en este caso tiene un porcentaje, considerando los cuatro tipos de clase, con mayor preferencia a distancias cercas y mayores tamaños tal y como sucedía en el Escenario 2, de $80,75\% \pm 9,95\%$, superando al segundo mejor caso que es el YOLOv26 del Escenario 2 (véase Figura 4.16), con $84,25\% \pm 23,64\%$, es decir, el promedio del YOLOv26 en el Escenario 2 es un 4% mejor aproximadamente, pero con una desviación típica de los resultados de 23,64% frente al 9,95% del YOLOv11 de este Escenario 3.

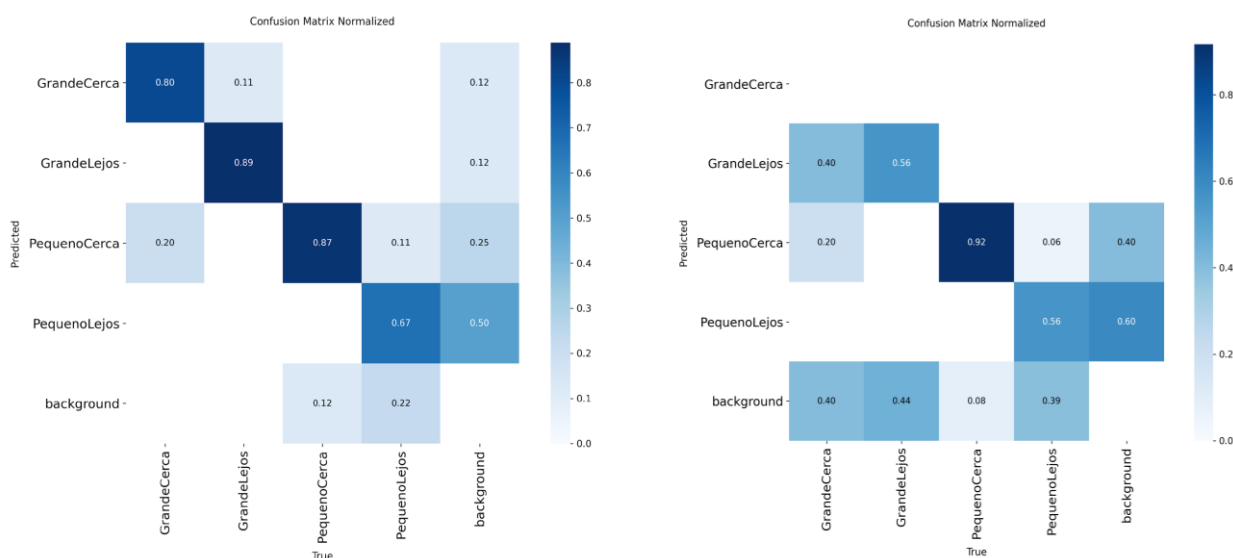


Figura 4.29 Matrices de confusión normalizadas de YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) del Escenario 3 (Elaboración propia)

En cuanto a las imágenes ‘*PRED*’ y ‘*LABEL*’, los resultados visuales (véase Figura 4.30 y Figura 4.31) muestran que el modelo es capaz de adaptarse bien a ambos tipos de espectro. Sin embargo, las predicciones generadas por YOLOv26 son, en general, más bajas que el YOLOv11, tal y como se ha comprobado anteriormente de forma cuantitativa. El modelo YOLOv11 muestra una mejor capacidad para detectar objetos tanto en imágenes térmicas como visibles, lo que sugiere una mayor capacidad de generalización.

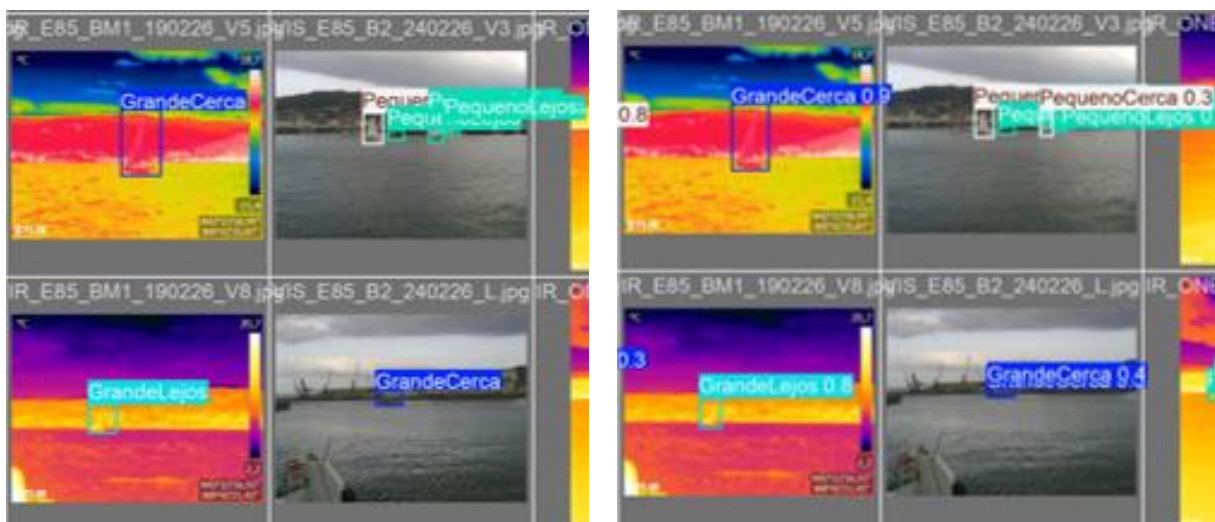


Figura 4.30 Ejemplos de imágenes ‘*LABEL*’ (a la izquierda) y ‘*PRED*’ (a la derecha) del modelo YOLOv11 en el Escenario 3 (Elaboración propia)

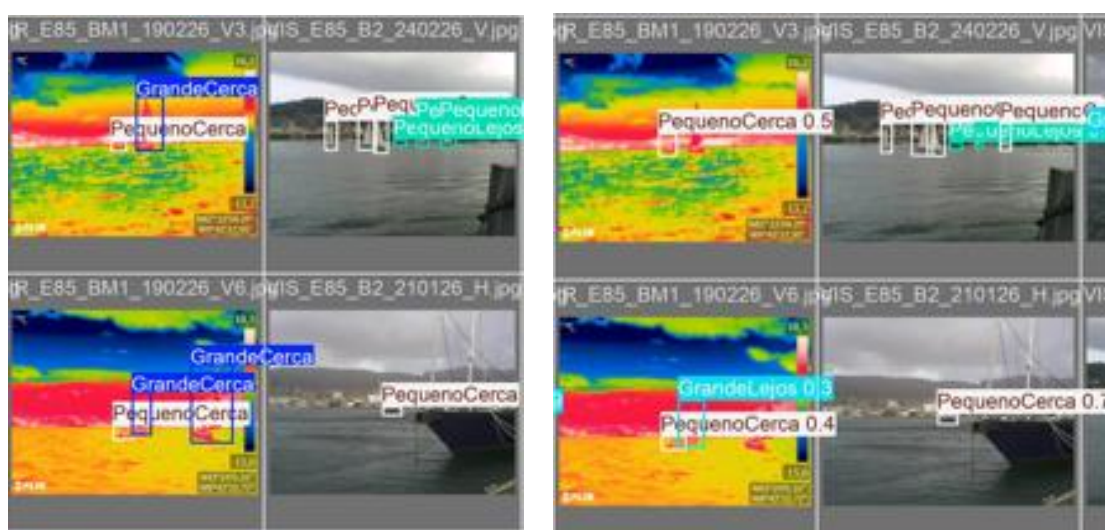


Figura 4.31 Ejemplos de imágenes ‘*LABEL*’ (a la izquierda) y ‘*PRED*’ (a la derecha) del modelo YOLOv26 en el Escenario 3 (Elaboración propia)

Las imágenes sometidas al proceso de testeo demuestran que el enriquecimiento del modelo con la introducción de imágenes tanto térmicas como visibles, en vez de poder causar confusión a los modelos, mantiene o incluso aumenta la capacidad general de detección. Nótese la necesidad de aumentar el número de datos de entrada, sobre todo en las visibles, donde la imagen visible superior uno de los dos objetos no lo detecta, y en las imágenes térmicas de la FLIR One Edge todas son detectadas, aunque erróneamente clasificado el objeto de la imagen superior (véase Figura 4.32).

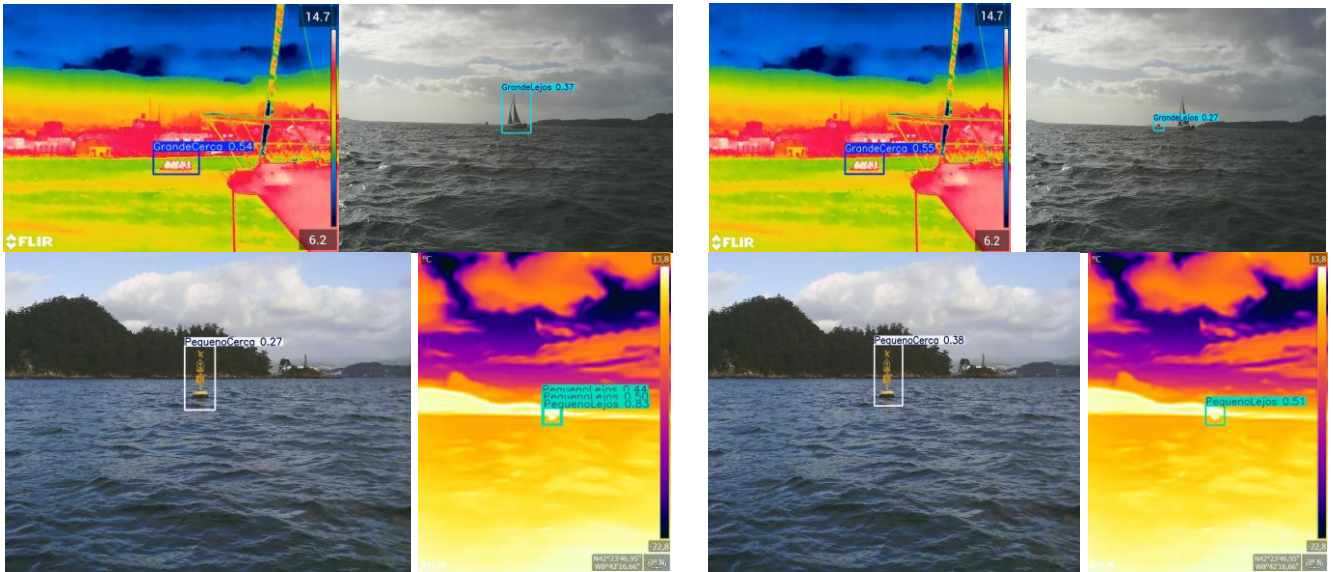


Figura 4.32 Resultados de la fase de testeo del Escenario 3 del modelo YOLOv11 (a la izquierda) y YOLOv26 (a la derecha) (Elaboración propia)

Mask R-CNN: para este caso de estudio, el conjunto de datos se dividió de la siguiente manera:

- 100 imágenes para el aprendizaje, divididas manualmente en 80 imágenes de entrenamiento y 20 de validación. Entre las 80 de entrenamiento se incluyen 30 imágenes de la FLIR E-85, 25 de la FLIR One Edge y 25 visibles, mientras que el conjunto de validación cuenta con 7 imágenes tanto de la FLIR E-85 como de la FLIR One Edge y 6 imágenes visibles.
- 20 imágenes de testeo usadas para el modelo YOLO.

Con respecto a las pérdidas del proceso de entrenamiento, se alcanzan valores muy favorables en el caso de la clasificación, como se muestran en las gráficas de la Figura 4.33 y la Figura 4.34, con valores que rápidamente tienden a 0.



Figura 4.33 Gráficas de pérdidas del modelo Mask R-CNN durante la fase de entrenamiento en el Escenario 3 (Elaboración propia)

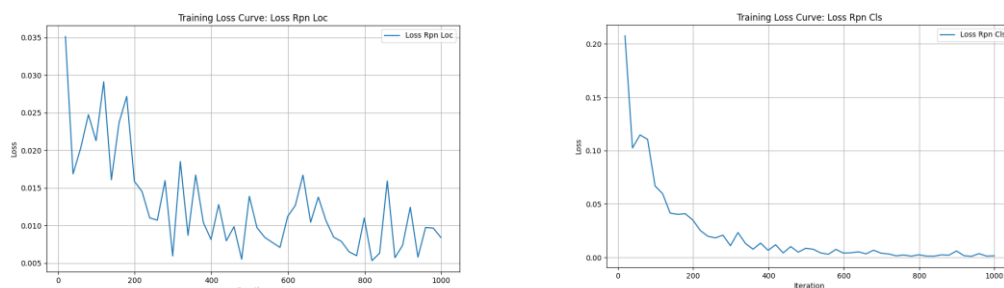


Figura 4.34 Gráficas de pérdidas asociadas a la RPN en el Escenario 3 (Elaboración propia)

Según las métricas de validación que se muestran en la Figura 4.35 y la Figura 4.36, tanto las correspondientes a la *bounding box* como a la *segmentation*, presentan valores ligeramente inferiores con respecto a los escenarios antes estudiados, con 30% (tendencia 35%), 60% (tendencia 70%) y 32% (tendencia 30%) en detección para *AP*, *AP50* y *AP75*, respectivamente; y 30% (tendencia 35%), 50% (tendencia 65%) y 32% (tendencia 25%) en segmentación para *AP*, *AP50* y *AP75*, respectivamente. Al duplicar de espectro, los modelos de *Deep Learning* tienen tendencia a mejorar sus prestaciones debido al enriquecimiento de los bancos de entrada, de ahí el mejor resultado en el YOLOv11 en el Escenario 3 con respecto a los YOLO de los otros escenarios. No obstante, ese enriquecimiento se recomienda que vaya vinculado a un aumento del número de datos, sobre todo en modelos más complejos con Mask R-CNN, de ahí su ligero empeoramiento en su rendimiento en este Escenario 3.

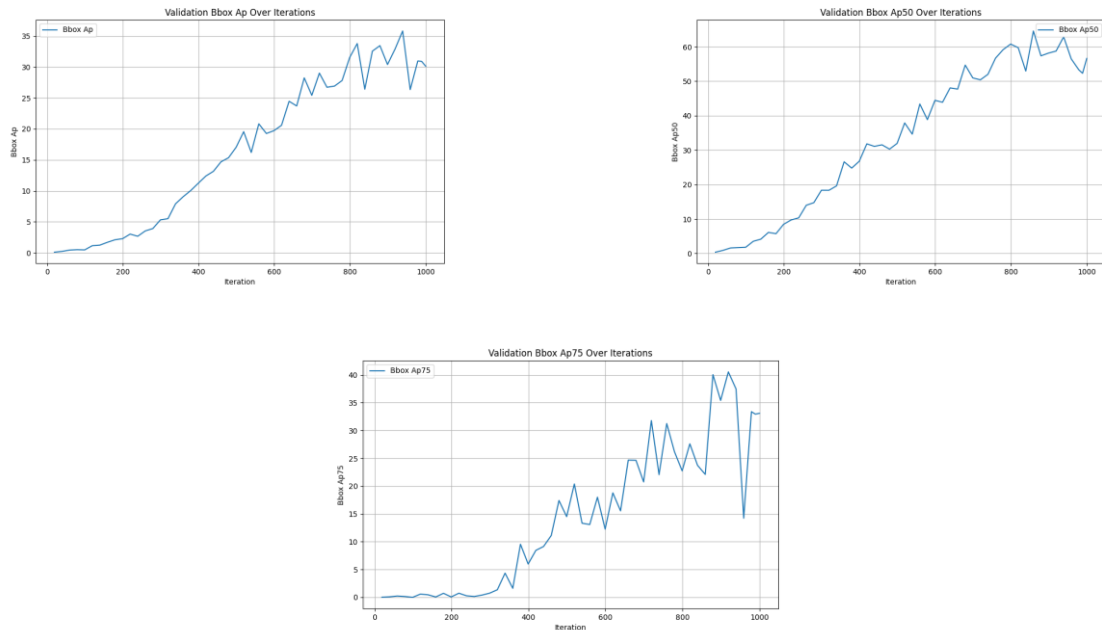


Figura 4.35 Métricas de evaluación del *bounding box* del conjunto de validación del Escenario 3 (Elaboración propia)

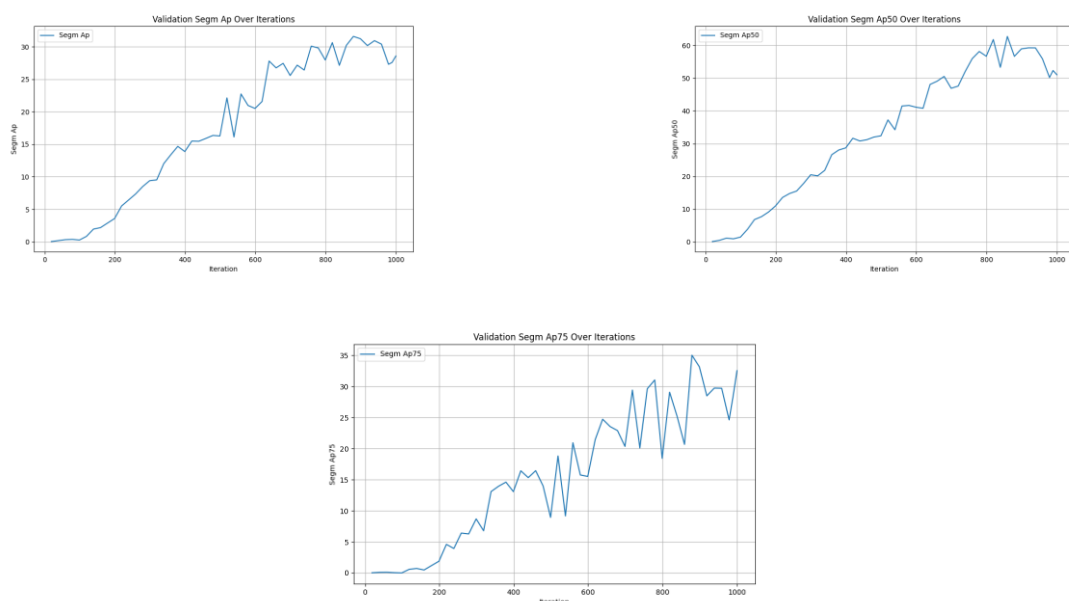


Figura 4.36 Métricas de evaluación de la segmentación del conjunto de validación del Escenario 3 (Elaboración propia)

En la Figura 4.37 se aprecia como sigue existiendo un gran número de falsos positivos en la zona de ‘background’, siendo este comportamiento más acusado que en los escenarios anteriores. En este caso, aparece una mayor variabilidad en las clasificaciones, así como la no detección de objetos etiquetados como ‘PequenoLejos’, aunque en el resto de las clases llega a un 50% de aciertos, lo que se traduce como en el Escenario 2, el modelo detecta mejor objetos más cercanos que lejanos.

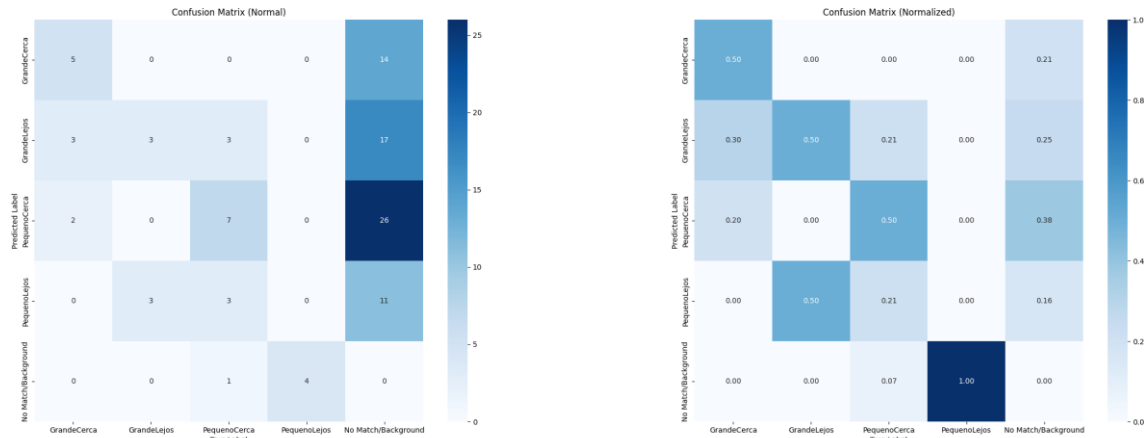


Figura 4.37 Matriz de confusión normal (a la izquierda) y normalizada (a la derecha) del modelo Mask R-CNN en el Escenario 3 (Elaboración propia)

A pesar de todo ello, los siguientes resultados visuales ponen de manifiesto la capacidad de Mask R-CNN de generar segmentaciones detalladas. Además, de forma generalizada se han reducido los errores debido a la superposición de las máscaras, ayudado por el enriquecimiento de los datos de entrada. En la Figura 4.38 se observan algunos de los escenarios más complejos en los que el modelo rinde de forma adecuada en cuanto a la localización, detección y clasificación de los objetos.

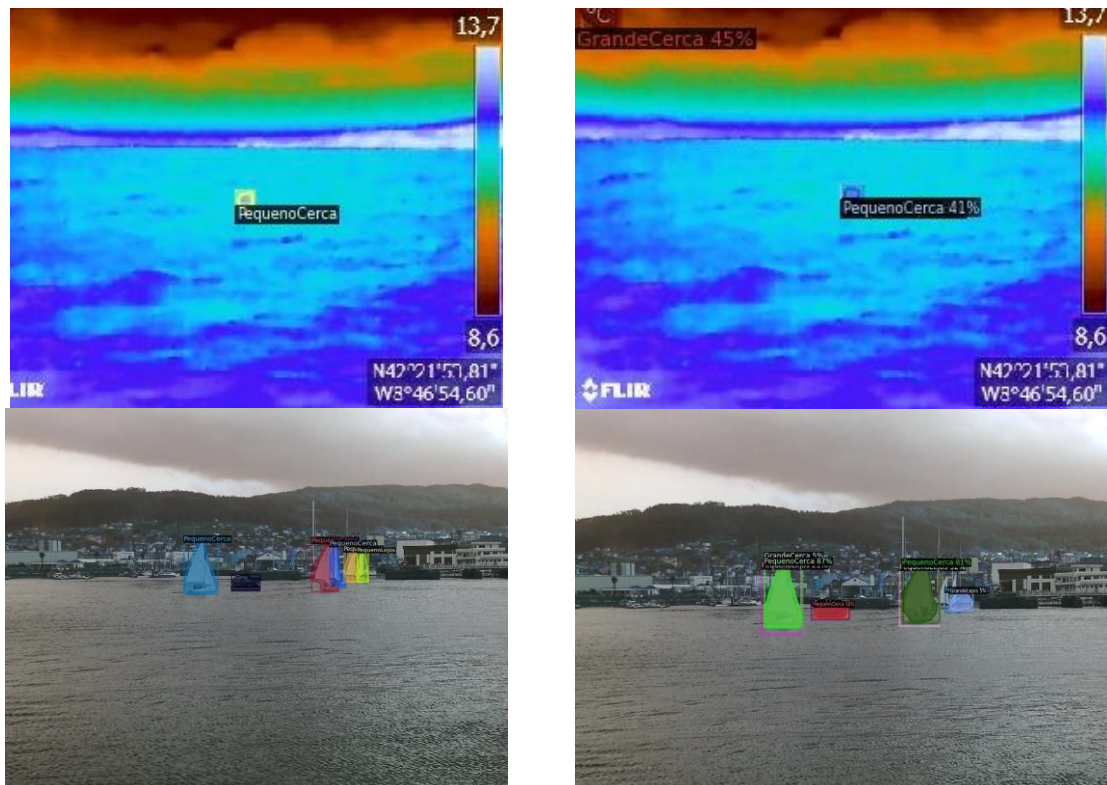


Figura 4.38 Ejemplos de imágenes ‘LABEL’ (a la izquierda) y ‘PRED’ (a la derecha) del modelo Mask R-CNN en el Escenario 3 (Elaboración propia)

Por último, en el proceso de inferencia de las 20 imágenes se comprueba que, tras el enriquecimiento del modelo, este ha sido capaz de lograr buenas detecciones, como puede verse entre los ejemplos de la Figura 4.39, con todo acertado menos un falso positivo debido a una marca de agua de la primera imagen térmica. El análisis de este escenario pone de manifiesto que la combinación de imágenes térmicas y visibles introduce una mayor complejidad en el proceso de aprendizaje, afectando especialmente a la estabilidad y consistencia de los modelos. No obstante, también permite mejorar la detección en aquellos casos donde la información térmica resulta limitada.

Entre los modelos evaluados, YOLOv11 y YOLOv26 destaca por su mayor robustez y capacidad de adaptación a datos heterogéneos, sobre todo el YOLOv11, mientras que Mask R-CNN ofrece una segmentación más detallada, aunque más sensible a la calidad de las imágenes y con mayor necesidad del aumento del número de datos entrada dado la complejidad del modelo. Este trabajo por tanto evidencia el potencial de la combinación de diferentes fuentes de información, térmica y visual, así como la necesidad de modelos capaces de adaptarse a entornos complejos y variables, requiriendo bancos de datos de entrada grandes para alcanzar mayor precisión en la detección (segmentación).

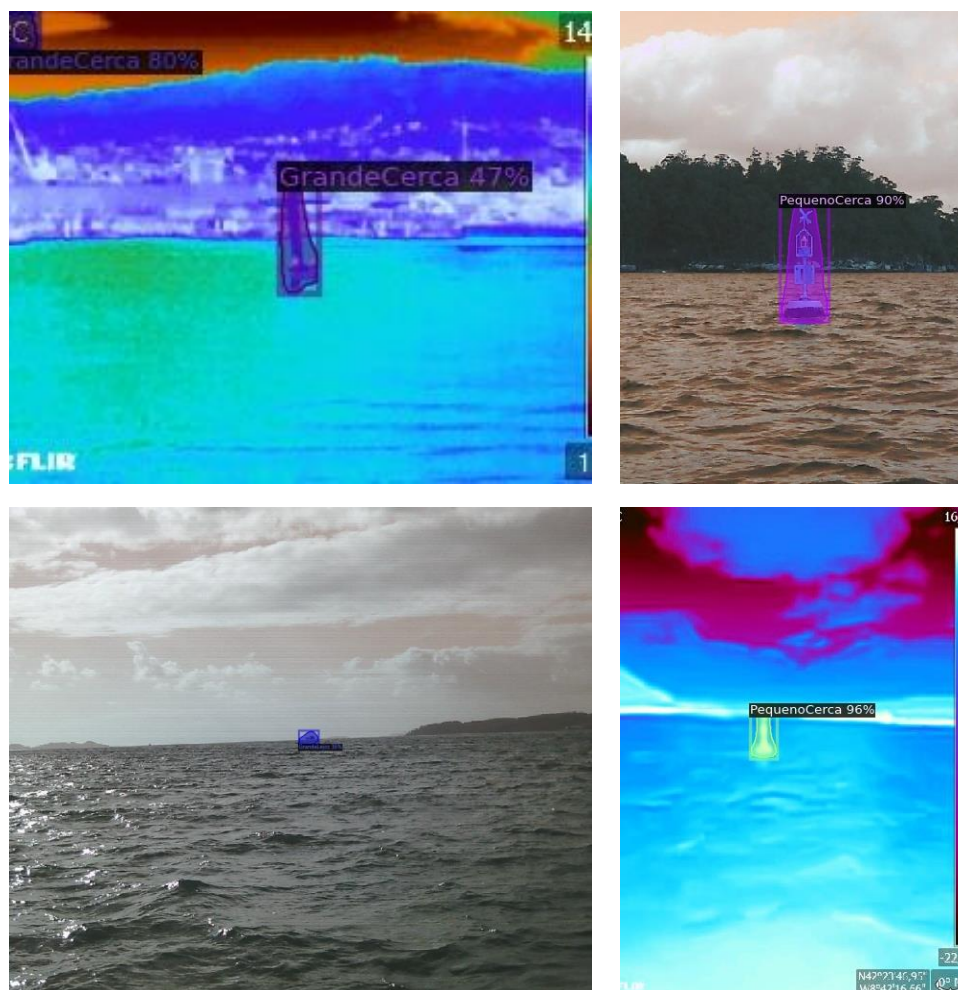


Figura 4.39 Resultados de la fase de testeo del modelo Mask R-CNN en el Escenario 3 (Elaboración propia)

4.4 Análisis y comparaciones

El análisis conjunto de los resultados obtenidos en los tres escenarios permite evaluar el comportamiento de los modelos en función de la complejidad del conjunto de datos y del tipo de imágenes empleadas, así como establecer una comparación directa entre los modelos YOLOv11, YOLOv26 y Mask R-CNN en condiciones progresivamente más exigentes. En primer lugar, en los tres

escenarios planteados se observa una convergencia adecuada durante el entrenamiento, con valores de pérdidas generalmente inferiores a la unidad, lo que indica que los modelos han sido capaces de aprender patrones representativos del conjunto de datos. No obstante, a medida que aumenta la complejidad del escenario (especialmente en los escenarios 2 y 3, donde se incorporan imágenes de menor calidad y del espectro visible) se aprecia una mayor variabilidad en las métricas, lo que refleja la dificultad añadida del problema y la necesidad de un mayor esfuerzo de generalización por parte de los modelos. A pesar de ello, el YOLOv26 en el Escenario 2, y sobre todo el YOLOv11 en el Escenario 3, mostraron una gran resiliencia e incluso mejores métricas, dado que se ha enriquecido la variedad de datos según calidad (alta gama con FLIR E-85 y bajo coste con FLIR One Edge) y espectro (térmica y visible), además que la arquitectura ligera del YOLO permite adaptarse con menores dificultades a dichos cambios. Al ser modelos más complejos las arquitecturas que a mayores permiten la segmentación, como es el caso del Mask R-CNN, estos modelos además de enriquecer los datos de entrada, el aumento de la cantidad del banco de datos es primordial para que dichas métricas suban porcentualmente. Entre las clases detectadas, se apreció una mejor tendencia en la identificación de los tamaños *Grandes* con respecto a los *Pequeños*, y con respecto a las distancias se obtienen mejores resultados en *Cerca* con respecto a *Lejos*. Esa tendencia era la esperada, ya que dichos modelos detectan mejor con el objeto rodeado del mismo fondo, en este caso la mar, y no fondo mezclado donde las cordilleras en los objetos situados hacia el fondo pueden representar una huella térmica semejante a los objetos de interés. Y a misma distancia, si la huella térmica del objeto es mayor, esto facilitará al modelo su detección. En las siguientes tablas (Tabla 4-1, Tabla 4-2, Tabla 4-3) se muestra un resumen de las comparaciones entre las métricas obtenidas en los distintos escenarios.

Modelo	Métricas globales	Rendimiento GC/GL/PC/PL	Observaciones
YOLOv11	Precision: 80% Recall: 60% mAP50: 70% mAP50-95: 40%	50% / 67% / 50% / 67%	Rendimiento ligeramente mejor en la v26 que en v11 Indiferencia respecto al tamaño del objeto detectado Mejor prestación en objetos a larga distancia
YOLOv26	Precision: 80% Recall: 60% mAP50: 70% mAP50-95: 40%	50% / 75% / 50% / 60%	
Mask R-CNN	Det: AP 35% AP50 70% AP50-95 30%	43% / 29% / 25% / 60%	Peor rendimiento con respecto a YOLO debido a mayor complejidad Detecta bien objetos “PequenoLejos”
	Seg: AP 35% AP50 65% AP75 25%		

Tabla 4-1 Resumen comparativo de las métricas en el conjunto de validación del Escenario 1 (Elaboración propia)

Modelo	Métricas globales	Rendimiento GC/GL/PC/PL	Observaciones
YOLOv11	<i>Precision</i> : 60% <i>Recall</i> : 60% mAP50: 60% mAP50-95: 30%	83% / 40% / 53% / 40%	Aumentan los falsos positivos por la cámara de menor resolución Baja la métrica <i>precision</i> y mAP
YOLOv26	Valores mantenidos respecto al Escenario 1	100% / 100% / 87% / 50%	Gran robustez. Incrementos generales en rendimiento por clase. Ligeramente sensible al tamaño y distancias (mejores resultados en objetos grandes y cercanos)
Mask R-CNN	Det: Valores mantenidos respecto al Escenario 1	60% / 30% / 78% / 33%	Incremento en predicciones “background” Mejora notable en la detección de clases “Cerca”
	Seg: Valores mantenidos respecto al Escenario 1		

Tabla 4-2 Resumen comparativo de las métricas en el conjunto de validación del Escenario 2 (Elaboración propia)

Modelo	Métricas globales	Rendimiento GC/GL/PC/PL	Observaciones
YOLOv11	<i>Precision</i> : 80% <i>Recall</i> : 80% mAP50: 80% mAP50-95: >40%	80% / 89% / 87% / 67%	Modelo más robusto (valores del 80%) Gran mejora por enriquecimiento de datos Preferencia a distancias cortas y tamaños grandes
YOLOv26	<i>Precision</i> : 60% <i>Recall</i> : 70% mAP50: >70% mAP50-95: 40%	0% / 56% / 92% / 56%	Menos consistente que la v11 en este escenario
Mask R-CNN	Det: AP 30% AP50 60% AP75 32%	50% en general 0% en PL	Ligero empeoramiento Requiere aumentar el volumen de datos No detecta “PequenoLejos”
	Seg: AP 30% AP50 50% AP75 32%		

Tabla 4-3 Resumen comparativo de las métricas en el conjunto de validación del Escenario 3 (Elaboración propia)

En cuanto al comportamiento general, uno de los aspectos más destacables es la dificultad común de los modelos para discriminar objetos frente al oleaje, como se observa en la Figura 4.40. Este fenómeno introduce ruido en la escena y puede dar lugar a falsas detecciones o a la pérdida de objetos. Otro aspecto relevante es la dificultad para reconocer correctamente estructuras específicas, como en los veleros, cuya geometría y elementos característicos (mástiles, velas) no siempre son captados de forma consistente por los modelos, especialmente en imágenes térmicas. Esta limitación pone de manifiesto la dependencia del modelo respecto a la calidad y representatividad del conjunto de datos utilizado.

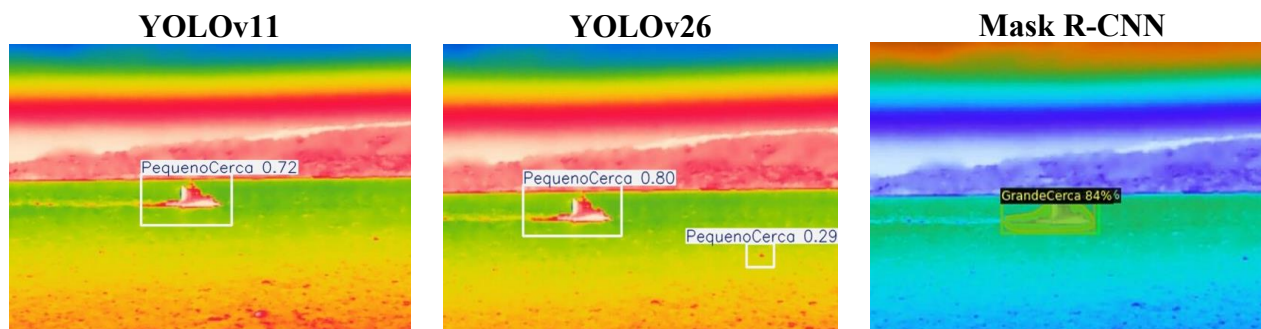


Figura 4.40 Ejemplo de detecciones en la misma imagen tras el proceso de testeo de los tres modelos (Elaboración propia)

En cuanto al tipo de imagen, como se puede observar en la Figura 4.41, se evidencia que los modelos funcionan de manera más eficaz sobre imágenes térmicas, mientras que en imágenes del espectro visible se observa una mayor dificultad en la detección, con un aumento de errores y una menor consistencia en las predicciones (la clase real del objeto en la figura mencionada es ‘GrandeCerca’), dado que el espectro térmico es más resiliente a condiciones climáticas adversas y a pesar que los modelos venían preentrenados con banco de datos de imágenes visibles (banco de datos de *COCO*).

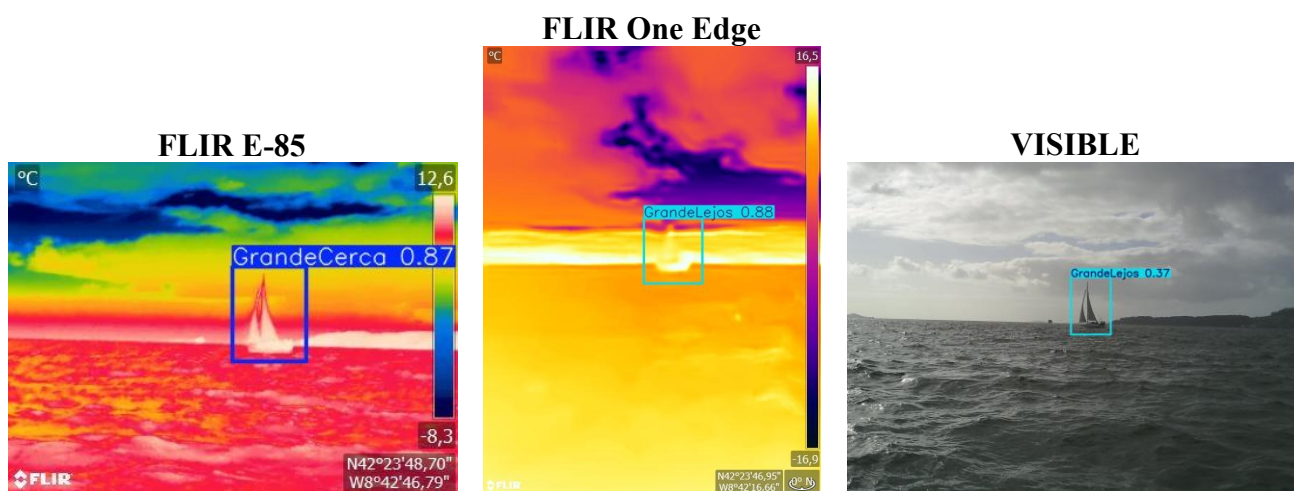


Figura 4.41 Comparativa de probabilidades de detección en imágenes de distintas fuentes mediante el modelo YOLO (Elaboración propia)

Es importante destacar que parte del análisis cualitativo realizado está sujeto a un cierto grado de subjetividad, ya que la clasificación de los objetos en términos de tamaño o distancia depende del criterio adoptado durante el estudio, lo que conlleva a un cierto número de errores. Asimismo, en la fase de testeo, las imágenes utilizadas no están etiquetadas, lo que impide una evaluación cuantitativa directa y obliga a basar el análisis en la inspección visual de los resultados obtenidos. Finalmente, el uso de conjuntos de datos cada vez más variados ha contribuido a un enriquecimiento progresivo del modelo, mejorando su capacidad de adaptación a distintas condiciones, aunque también incrementando la complejidad del problema.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Cumplimiento de los objetivos

El presente Trabajo Fin de Grado ha cumplido de forma satisfactoria el objetivo principal de desarrollar y evaluar un sistema de detección automatizada de objetos en entornos marinos mediante el uso de imágenes termográficas y modelos de *Deep Learning*. Los resultados obtenidos demuestran que es posible aplicar este tipo de técnicas para identificar y localizar objetos de interés en condiciones reales, validando así la viabilidad del enfoque propuesto.

Desde el punto de vista técnico, los modelos implementados han mostrado una correcta convergencia durante el aprendizaje, con una evolución coherente de las funciones de pérdida y métricas de evaluación. Esto indica que los modelos han sido capaces de aprender patrones representativos a partir del conjunto de datos disponible, incluso siendo este de tamaño limitado.

En la comparación entre modelos, se concluye que las arquitecturas de la familia YOLO ofrecen el mejor comportamiento global en este caso de estudio. Destacan por su equilibrio entre precisión, robustez y velocidad de inferencia, lo que los convierte en una solución especialmente adecuada para aplicaciones en tiempo real en el ámbito marítimo. En particular, YOLOv11 presenta las mejores prestaciones en los resultados frente a YOLOv26, demostrado en el escenario más complejo, es decir, juntando imágenes térmicas de alta gama (FLIR E-85) con imágenes térmicas adquiridas por la cámara de bajo coste (FLIR One Edge) y con imágenes visibles. Con ello, se aprecia que el enriquecimiento de la variedad de los datos de entrada es fundamental, llegando a obtener una precisión (número de aciertos en detecciones) y un *recall* (cobertura en detectar todos los objetos reales existentes) del 80%, y un promedio del $80,75\% \pm 9,95\%$ de acierto teniendo en cuenta las cuatro clases de objeto estudiadas: ‘GrandeCerca’, ‘GrandeLejos’, ‘PequenoCerca’ y ‘PequenoLejos’, con mayor preferencia a los objetos de gran tamaño y corta distancia. Por su parte, Mask R-CNN, aunque proporciona una segmentación más precisa del contorno de los objetos, presenta un rendimiento inferior en términos de detección global y una mayor sensibilidad a la variabilidad del conjunto de datos, siendo fundamental para estos modelos más complejos la necesidad de aumentar el número de datos de entrada además de fomentar el enriquecimiento de la variedad de esos datos.

Asimismo, se ha comprobado que el uso de imágenes termográficas constituye una ventaja significativa frente al espectro visible, permitiendo mantener una capacidad de detección más consistente incluso en condiciones adversas o con menor calidad de imagen. Este resultado refuerza el interés de la termografía infrarroja como herramienta en aplicaciones de vigilancia y seguridad marítima.

No obstante, el sistema presenta ciertas limitaciones que condicionan su rendimiento. Entre ellas destacan la dificultad para detectar objetos pequeños o a gran distancia, la confusión con elementos del entorno y la dependencia del modelo respecto a la calidad, cantidad y equilibrio del conjunto de datos.

Además, la definición de las clases empleadas (grande/pequeño, cerca/lejos) introduce un cierto grado de subjetividad que puede influir en la evaluación de los resultados.

En conjunto, los resultados obtenidos permiten afirmar que se han alcanzado los objetivos planteados, demostrando el potencial de las técnicas de *Deep Learning* aplicadas a la detección de objetos en entornos marítimos mediante imágenes térmicas, en un ámbito en el que existe aún una limitada presencia de estudios específicos, lo que confiere a este trabajo un carácter pionero.

5.2 Líneas futuras de investigación

A partir de los resultados obtenidos, se identifican diversas líneas de trabajo que permitirían mejorar y ampliar el sistema desarrollado.

En primer lugar, resulta fundamental aumentar el tamaño y la diversidad del conjunto de datos, incorporando un mayor número de imágenes, diferentes condiciones ambientales, con mayor variabilidad de escenarios y clases de objetos. Esto permitiría mejorar la capacidad de generalización de los modelos y reducir errores en situaciones complejas.

Asimismo, se propone explorar otras arquitecturas de *Deep Learning*, como las mencionadas en el estado del arte, incluyendo modelos más avanzados de detección o enfoques basados en *transformers* (por ejemplo, DETR o RT-DETR), con el objetivo de analizar posibles mejoras en el rendimiento frente a las arquitecturas empleadas en este trabajo.

Otra línea de investigación relevante es la incorporación de información procedente de otros espectros o sensores, como imágenes visibles, radiogramas u otros sistemas de adquisición, lo que permitiría desarrollar sistemas multisensor más robustos mediante técnicas de fusión de datos.

Además, sería de gran interés avanzar hacia la implementación del sistema en tiempo real en plataformas embarcadas, optimizando los modelos para su ejecución en dispositivos con recursos limitados y evaluando su comportamiento en condiciones operativas reales.

Finalmente, se plantea la posibilidad de integrar funcionalidades adicionales como el seguimiento de objetos (*tracking*) o la predicción de trayectorias, lo que ampliaría significativamente el alcance del sistema en aplicaciones de vigilancia marítima, seguridad naval y apoyo a operaciones de búsqueda y rescate.

6 BIBLIOGRAFÍA

- [1] «Transporte marítimo: UNCTAD publica nuevas estadísticas | ONU Comercio y Desarrollo (UNCTAD)». Accedido: 23 de enero de 2026. [En línea]. Disponible en: <https://unctad.org/es/news/transporte-maritimo-unctad-publica-nuevas-estadisticas>
- [2] «La nueva Estrategia de Seguridad Marítima de Europa 2023», 2023.
- [3] «Maritime Security». Accedido: 23 de enero de 2026. [En línea]. Disponible en: <https://www.imo.org/en/ourwork/security/pages/default.aspx>
- [4] Y. Saba, J. Taha, J. Saul, y Y. Saba, «Three tankers damaged in Gulf and one seafarer killed as US-Iran conflict escalates», *Reuters*, 1 de marzo de 2026. Accedido: 9 de marzo de 2026. [En línea]. Disponible en: <https://www.reuters.com/business/energy/three-tankers-damaged-gulf-us-iran-conflict-escalates-2026-03-01/>
- [5] K. Talpur, R. Hasan, I. Gocer, S. Ahmad, y Z. Bhuiyan, «AI in Maritime Security: Applications, Challenges, Future Directions, and Key Data Sources», *Information*, vol. 16, n.º 8, p. 658, ago. 2025, doi: 10.3390/info16080658.
- [6] «El mar como conector global», El Orden Mundial - EOM. Accedido: 23 de enero de 2026. [En línea]. Disponible en: <https://elordenmundial.com/podcasts/no-es-el-fin-del-mundo/el-mar-como-conector-global/>
- [7] Armada, «Seguridad Marítima - La Armada, un actor indispensable - Especiales - Armada - Ministerio de Defensa - Gobierno de España». Accedido: 23 de enero de 2026. [En línea]. Disponible en: <https://armada.defensa.gob.es/ArmadaPortal/page/Portal/ArmadaEspannola/conocenosespeciales/prefLang-es/05actividades--seguridadmaritima>
- [8] «folleto-expo-seg-mari».
- [9] «COVAM-Informe_Incidentes_Seguridad_Maritima.pdf». Accedido: 23 de enero de 2026. [En línea]. Disponible en: https://encomar.covam.es/documents/20182/395165/COVAM-Informe_Incidentes_Seguridad_Maritima.pdf?utm_source=chatgpt.com
- [10] «Disposición 21884 del BOE núm. 314 de 2021». Accedido: 16 de febrero de 2026. [En línea]. Disponible en: <https://www.boe.es/boe/dias/2021/12/31/pdfs/BOE-A-2021-21884.pdf>
- [11] «Disposición 10762 del BOE núm. 130 de 2024». Accedido: 16 de febrero de 2026. [En línea]. Disponible en: <https://www.boe.es/boe/dias/2024/05/29/pdfs/BOE-A-2024-10762.pdf>
- [12] Armada, «Seguridad Marítima - La Armada, un actor indispensable - Especiales - Armada - Ministerio de Defensa - Gobierno de España». Accedido: 9 de marzo de 2026. [En línea]. Disponible en: <https://armada.defensa.gob.es/ArmadaPortal/page/Portal/ArmadaEspannola/conocenosespeciales/prefLang-es/05actividades--seguridadmaritima>

- [13] «Radar Technology Overview: Types, Principles, and Applications Across Domains | Lakeda Radar - Advanced Security and Surveillance Radar Solutions». Accedido: 23 de enero de 2026. [En línea]. Disponible en: https://www.lakedaradar.com/en/news/radar-technology-types-principle-and-applications/?utm_source=chatgpt.com
- [14] I. S. de Navegación, «El Radar | Instituto de Navegación». Accedido: 23 de enero de 2026. [En línea]. Disponible en: <https://www.isndf.com.ar/el-radar/>
- [15] Dataintel, «Maritime Surveillance Radar Market Research Report 2033». Accedido: 23 de enero de 2026. [En línea]. Disponible en: <https://dataintel.com/report/maritime-surveillance-radar-market>
- [16] «Transpondedores AIS». Accedido: 23 de enero de 2026. [En línea]. Disponible en: <https://www.imo.org/es/ourwork/safety/pages/ais.aspx>
- [17] alejandrog183, «¿COMO FUNCIONA EL A.I.S.?», *electromaritima*. Accedido: 23 de enero de 2026. [En línea]. Disponible en: <https://www.electromaritima.com.mx/post/como-funciona-el-a-i-s>
- [18] International Maritime Organization, *GMDSS MANUAL*, 2024 Edition. 2024. [En línea]. Disponible en: <https://doi.org/10.62454/KJ970E>
- [19] «Facultat de Nàutica de Barcelona. | FNB». Accedido: 23 de enero de 2026. [En línea]. Disponible en: <https://www.fnb.upc.edu/>
- [20] «Elementos que componen la Cámara Fotográfica». Accedido: 23 de enero de 2026. [En línea]. Disponible en: <https://www.fotonostra.com/fotografia/elementoscamera>
- [21] N. Wawrzyniak, T. Hyla, y A. Popik, «Vessel Detection and Tracking Method Based on Video Surveillance», *Sensors*, vol. 19, n.º 23, p. 5230, nov. 2019, doi: 10.3390/s19235230.
- [22] «Cámaras de seguridad visibles o térmicas: ¿cuál necesita? | Flir». Accedido: 5 de febrero de 2026. [En línea]. Disponible en: <https://www.flir.com/es-mx/discover/security/visible-vs-thermal-security-cameras/>
- [23] «Cámaras Termográficas: Cómo funcionan y por qué pueden ser necesarias». Accedido: 5 de febrero de 2026. [En línea]. Disponible en: <https://www.promax.es/esp/noticias/400/camara-termografica-como-funcionan-y-por-que-pueden-ser-necesarias/>
- [24] A. Harati-Mokhtari, A. Wall, P. Brooks, y J. Wang, «Automatic Identification System (AIS): Data Reliability and Human Error Implications», *J. Navig.*, vol. 60, n.º 3, pp. 373-389, sep. 2007, doi: 10.1017/S0373463307004298.
- [25] H. Ingeniería, «¿Qué es la termografía? ¿Para qué sirve la termografía?», *Hume Ingeniería*. Accedido: 5 de febrero de 2026. [En línea]. Disponible en: <https://humeingenieria.es/blog/consejos-sobre-humedades/tu-primer-contacto-con-la-termografia/>
- [26] «Fundamentos de la termografía infrarroja», *Power-MI*. Accedido: 5 de febrero de 2026. [En línea]. Disponible en: <https://power-mi.com/es/content/fundamentos-de-la-termograf%C3%ADa-infrarroja>
- [27] «Descubre cómo funcionan las cámaras térmicas». Accedido: 5 de febrero de 2026. [En línea]. Disponible en: <https://www.apliter.com/blog/descubre-como-funcionan-cameras-termicas/>
- [28] «Ley de Planck», *Wikipedia, la enciclopedia libre*. 27 de agosto de 2025. Accedido: 5 de febrero de 2026. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Ley_de_Planck&oldid=169205914
- [29] Z. Dukowitz, «Thermographic Inspection: An In-Depth Guide [New for 2026]», *MFE Inspection Solutions*. Accedido: 5 de febrero de 2026. [En línea]. Disponible en: <https://mfe-is.com/thermographic-inspection/>
- [30] «MA_0021_ES.pdf». Accedido: 5 de febrero de 2026. [En línea]. Disponible en: http://www.flirmedia.com/MMC/CSV/Maritime/MA_0021_ES.pdf
- [31] L. Tadić, I. Golub Medvešek, I. Vujović, y J. Šoda, «Infrared Thermography in Maritime Systems: A Systematic Review», *Appl. Sci.*, vol. 15, n.º 23, p. 12551, nov. 2025, doi: 10.3390/app152312551.

- [32] J. Brauchle, S. Bayer, D. Hein, R. Berger, y S. Pless, «MACS-Mar: a real-time remote sensing system for maritime security applications», *CEAS Space J.*, vol. 11, n.º 1, pp. 35-44, mar. 2019, doi: 10.1007/s12567-018-0207-7.
- [33] *Inteligencia artificial: 101 cosas que debes saber hoy sobre nuestro futuro*, Sexta edición. Barcelona: Alienta Editorial, 2023.
- [34] «Inteligencia Artificial como servicio: reconocimiento de imágenes». Accedido: 5 de febrero de 2026. [En línea]. Disponible en: <https://www.paradigmadigital.com/techbiz/inteligencia-artificial-servicio-reconocimiento-imagenes/>
- [35] O. J. Reyes Ortiz, M. Mejia, y J. S. Useche Castelblanco, «Técnicas de inteligencia artificial utilizadas en el procesamiento de imágenes y su aplicación en el análisis de pavimentos», *Rev. EIA*, vol. 16, n.º 31, pp. 189-207, ene. 2019, doi: 10.24050/reia.v16i31.1215.
- [36] T. M. Mitchell, *Machine learning*, Nachdr. en McGraw-Hill series in Computer Science. New York: McGraw-Hill, 2013.
- [37] W. Forero-Corba y F. N. Bennasar, «Técnicas y aplicaciones del Machine Learning e inteligencia artificial en educación: una revisión sistemática», *RIED-Rev. Iberoam. Educ. Distancia*, vol. 27, n.º 1, pp. 209-253, ene. 2024, doi: 10.5944/ried.27.1.37491.
- [38] «IA vs. machine learning vs. deep learning vs. redes neuronales | IBM». Accedido: 5 de febrero de 2026. [En línea]. Disponible en: <https://www.ibm.com/es-es/think/topics/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks>
- [39] «Redes neuronales artificiales: qué son y cómo se entrenan | Blog UE», Universidad Europea. Accedido: 5 de febrero de 2026. [En línea]. Disponible en: <https://universidadeuropea.com/blog/redes-neuronales-artificiales/>
- [40] A. Cifuentes, E. Mendoza, M. Lizcano, y A. Santrich, «Desarrollo de una red neuronal convolucional para reconocer patrones en imágenes».
- [41] Y. Li, H. Su, C. R. Qi, N. Fish, D. Cohen-Or, y L. J. Guibas, «Joint embeddings of shapes and images via CNN image purification», *ACM Trans Graph*, vol. 34, n.º 6, p. 234:1-234:12, nov. 2015, doi: 10.1145/2816795.2818071.
- [42] S. G. Pajares, «Redes neuronales convolucionales. Aplicación a la detección de objetos.».
- [43] B. Selcuk y T. Serif, «A Comparison of YOLOv5 and YOLOv8 in the Context of Mobile UI Detection», en *Mobile Web and Intelligent Information Systems*, vol. 13977, M. Younas, I. Awan, y T.-M. Grønli, Eds., en *Lecture Notes in Computer Science*, vol. 13977, Cham: Springer Nature Switzerland, 2023, pp. 161-174. doi: 10.1007/978-3-031-39764-6_11.
- [44] «Máscara R-CNN Explicada: Guía, usos y YOLO | Ultralytics». Accedido: 9 de marzo de 2026. [En línea]. Disponible en: <https://www.ultralytics.com/es/blog/what-is-mask-r-cnn-and-how-does-it-work>
- [45] «How single-shot detector (SSD) works? | ArcGIS API for Python | Esri Developer». Accedido: 9 de marzo de 2026. [En línea]. Disponible en: <https://developers.arcgis.com/python/latest/guide/how-ssd-works/>
- [46] R. Keyes, «Zefs Guide to Deep Learning».
- [47] «You Only Look Once: Unified, Real-Time Object Detection». Accedido: 20 de febrero de 2026. [En línea]. Disponible en: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf
- [48] «R. Girshick, J. Donahue, T. Darrell and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 2014, pp. 580-587, doi: 10.1109/CVPR.2014.81. keywords: {Proposals;Feature extraction;Training;Visualization;Object detection;Vectors;Support vector machines},». Accedido: 20 de febrero de 2026. [En línea]. Disponible en: <https://arxiv.org/pdf/1311.2524>

- [49] «ACTAS DEL SIMPOSIO DE ROBÓTICA, BIOINGENIERÍA Y VISIÓN POR COMPUTADOR». Accedido: 5 de febrero de 2026. [En línea]. Disponible en: https://dspace.umh.es/bitstream/11000/32893/1/actas_Simposio%202024%20-%20eheredia.pdf
- [50] «Los Barcos de Guerra de Eugenio - Y-504». Accedido: 19 de febrero de 2026. [En línea]. Disponible en: https://www.losbarcosdeeukenio.com/barcos/es/es/ae_Y504.html#gsc.tab=0
- [51] «FLIR E85 WIFI 78502-0201», Cámaras Termográficas Flir. Accedido: 10 de febrero de 2026. [En línea]. Disponible en: <https://camarastermograficas.eu/cámaras-termográficas/212-flir-e85-wifi-78502-0201-flir.html>
- [52] «E85 Cámara termográfica avanzada», GMS Instruments. Accedido: 10 de febrero de 2026. [En línea]. Disponible en: <https://gms-instruments.com/es/product/e85-advanced-thermal-imaging-camera/>
- [53] «17-3307-BLD_ES.pdf». Accedido: 10 de febrero de 2026. [En línea]. Disponible en: http://www.flirmedia.com/MMC/THG/Brochures/17-3307/17-3307-BLD_ES.pdf
- [54] «Cámara termográfica FLIR Edge - Android y iOS». Accedido: 10 de febrero de 2026. [En línea]. Disponible en: <https://www.apliter.com/producto/flir-edge-camara-termografica-smartphone/>
- [55] «Flir Edge | Flir». Accedido: 10 de febrero de 2026. [En línea]. Disponible en: <https://www.flir.com/es-mx/products/flir-edge/>
- [56] A. Dutta y A. Zisserman, «The VIA Annotation Software for Images, Audio and Video», en *Proceedings of the 27th ACM International Conference on Multimedia*, Nice France: ACM, oct. 2019, pp. 2276-2279. doi: 10.1145/3343031.3350535.
- [57] «colab.google», colab.google. Accedido: 23 de febrero de 2026. [En línea]. Disponible en: <http://0.0.0.0:8080/>
- [58] M. L. Ali y Z. Zhang, «The YOLO Framework: A Comprehensive Review of Evolution, Applications, and Benchmarks in Object Detection».
- [59] T. Diwan, G. Anirudh, y J. V. Tembhurne, «Object detection using YOLO: challenges, architectural successors, datasets and applications», *Multimed. Tools Appl.*, vol. 82, n.º 6, pp. 9243-9275, mar. 2023, doi: 10.1007/s11042-022-13644-y.
- [60] M. R. Munawar, «How do Hyperparameters of YOLOv5 Work?», Medium. Accedido: 10 de marzo de 2026. [En línea]. Disponible en: <https://muhammadrizwanmunawar.medium.com/how-hyperparameters-of-yolov5-works-ec4d25f311a2>
- [61] Ultralytics, «Ultralytics YOLO26». Accedido: 9 de marzo de 2026. [En línea]. Disponible en: <https://docs.ultralytics.com/models/yolo26/>
- [62] «Mask R-CNN». Accedido: 4 de marzo de 2026. [En línea]. Disponible en: https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf
- [63] J. Hui, «Image segmentation with Mask R-CNN», Medium. Accedido: 4 de marzo de 2026. [En línea]. Disponible en: <https://jonathan-hui.medium.com/image-segmentation-with-mask-r-cnn-ebe6d793272>
- [64] Evan, *EdjeElectronics/Train-and-Deploy-YOLO-Models*. (22 de febrero de 2026). Jupyter Notebook. Accedido: 1 de marzo de 2026. [En línea]. Disponible en: <https://github.com/EdjeElectronics/Train-and-Deploy-YOLO-Models>
- [65] «Google Colab». Accedido: 12 de marzo de 2026. [En línea]. Disponible en: <https://colab.research.google.com/drive/1toMdtcLlkuWjhn1uDPPrHzSluN0r0IE0L>

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

ASPECTOS AMBIENTALES

Desde el punto de vista ambiental, el sistema desarrollado presenta un impacto reducido sobre el entorno, ya que la adquisición de imágenes termográficas se basa en una técnica pasiva que no implica la emisión de energía hacia el medio. A diferencia de otros sistemas de detección activos, como determinados radares, la captación de radiación infrarroja aprovecha la energía térmica emitida de forma natural por los objetos, minimizando posibles interferencias con el ecosistema marino. Esta característica resulta especialmente relevante en entornos sensibles, donde la reducción de perturbaciones externas constituye un criterio fundamental en el diseño y selección de tecnologías de vigilancia.

La mejora en la capacidad de detección temprana de embarcaciones, objetos flotantes o elementos a la deriva puede contribuir de manera indirecta a la prevención de accidentes marítimos. La anticipación de este tipo de situaciones reduce el riesgo de colisiones, varadas o incidentes que puedan derivar en vertidos contaminantes o daños al medio ambiente. En este sentido, el sistema propuesto no solo actúa como herramienta de seguridad, sino que también puede desempeñar un papel preventivo frente a impactos ambientales asociados a la actividad marítima.

Por otra parte, una vigilancia más precisa permite optimizar desplazamientos y operaciones, lo que puede traducirse en un uso más racional de recursos energéticos, contribuyendo así a una menor huella energética. Aunque el presente trabajo no aborda específicamente aplicaciones de monitorización ambiental, la tecnología desarrollada podría adaptarse a tareas de detección de anomalías térmicas relacionadas con focos de contaminación, vertidos o alteraciones del entorno. Asimismo, desde una perspectiva de sostenibilidad tecnológica, el empleo de modelos optimizados y hardware de bajo consumo favorece la eficiencia energética en caso de futura implementación embarcada.

ASPECTOS ECONÓMICOS

Para el ámbito económico, la incorporación de sistemas automáticos de detección basados en inteligencia artificial puede suponer una optimización significativa de los recursos operativos. Una vigilancia más eficaz del entorno marítimo permite identificar con antelación situaciones potencialmente peligrosas, reduciendo la probabilidad de incidentes que conlleven elevados costes derivados de daños materiales, interrupciones del servicio o actuaciones de emergencia. En este sentido, la inversión en tecnologías de detección temprana puede traducirse en un ahorro económico a medio y largo plazo.

El trabajo desarrollado pone de manifiesto, además, la viabilidad económica de soluciones avanzadas basadas en *Deep Learning* sin necesidad de recurrir a infraestructuras de alto coste en fases iniciales. El uso de plataformas de computación en la nube y de modelos pre-entrenados demuestra que es posible llevar a cabo desarrollos complejos con recursos limitados, lo que facilita su adopción en entornos académicos, institucionales o incluso en organizaciones con presupuestos ajustados. Esta flexibilidad permite adaptar la solución a distintos niveles de inversión, desde sistemas portátiles de bajo coste hasta integraciones en plataformas profesionales de mayor complejidad.

Por tanto, el trabajo evidencia que la aplicación de técnicas de inteligencia artificial en el ámbito marítimo no solo es técnicamente viable, sino también económicamente asumible, favoreciendo su potencial escalabilidad y su transferencia a entornos reales de operación.

ASPECTOS INDUSTRIALES

En el ámbito industrial, el proyecto se enmarca claramente dentro del proceso de digitalización y modernización tecnológica del sector marítimo y naval. La combinación de sensores térmicos con algoritmos avanzados de aprendizaje profundo responde a una tendencia creciente hacia la automatización y el análisis inteligente de la información, especialmente en sistemas de vigilancia y seguridad.

El sistema propuesto es conceptualmente integrable en los sistemas electroópticos ya existentes, cada vez más habituales en sistemas de vigilancia naval. Su carácter modular y su enfoque basado en software permiten una adaptación progresiva a diferentes configuraciones industriales, facilitando su incorporación como complemento a sistemas tradicionales. Además, la aplicabilidad dual de la tecnología, tanto en el ámbito civil como en el militar, amplía su interés desde el punto de vista industrial y estratégico.

ASPECTOS SOCIALES

El trabajo está orientado a la mejora de la seguridad marítima mediante la detección automática de objetos y posibles personas en la mar empleando imágenes térmicas. En entornos donde las condiciones meteorológicas o la baja luminosidad limitan la percepción visual convencional, la incorporación de sistemas automáticos de detección constituye una herramienta de apoyo a la toma de decisiones, minimizando la fatiga del operador y aumentando la eficacia operativa en entornos exigentes.

La capacidad de detectar embarcaciones de pequeño porte, objetos flotantes o situaciones potencialmente peligrosas puede contribuir de forma directa a la protección de la vida humana, especialmente en contextos de vigilancia, patrulla o apoyo a operaciones de búsqueda y rescate. Asimismo, el desarrollo de tecnologías aplicadas a la seguridad marítima responde a una demanda social creciente de sistemas más eficaces, precisos y adaptados a entornos complejos.

ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

A lo largo del desarrollo del presente trabajo, se han abordado diversas reflexiones éticas y sociales que se han reflejado en las diferentes secciones descritas. Estas reflexiones han surgido como resultado del análisis crítico de las implicaciones éticas y sociales asociadas con el uso de la termografía infrarroja junto con sistemas de detección automatizada para la detección de objetos de interés en entornos marinos.

El sistema propuesto se enmarca en el ámbito de la vigilancia y la seguridad marítima, un contexto en el que la información generada puede influir directamente en la toma de decisiones operativas. Desde una perspectiva ética, resulta fundamental considerar que los modelos de detección automática empleados deben entenderse como herramientas de apoyo al operador humano y no como sistemas autónomos de decisión, por lo que la responsabilidad última sobre las decisiones adoptadas debe recaer siempre en personal cualificado, especialmente cuando están en juego la seguridad de personas o la integridad de bienes materiales.

La fiabilidad de los resultados obtenidos por los modelos desarrollados depende en gran medida de la calidad y representatividad del conjunto de datos empleado durante las fases de entrenamiento y la validación. La existencia de errores de detección ya sea en forma de falsas alarmas o de omisiones, pone de manifiesto una limitación inherente a este tipo de tecnologías. Desde el punto de vista ético y profesional, resulta imprescindible que estas limitaciones sean conocidas y comprendidas por los usuarios del sistema, evitando una confianza ciega en los resultados generados por la inteligencia artificial. Cualquier aplicación operativa futura debería contemplar mecanismos de supervisión humana, validación continua y revisión periódica del rendimiento del modelo.

En conjunto, el desarrollo del presente trabajo pone de manifiesto la importancia de integrar la reflexión ética y social en el diseño y evaluación de sistemas tecnológicos avanzados, de modo que la capacidad de manejar datos y conceptos complejos debe ir siempre acompañada de un juicio crítico sobre las consecuencias de su aplicación.

ANEXO III: CÓDIGO YOLO v11 Y v26

En el presente anexo se recogen las capturas del código utilizado para el entrenamiento, validación y prueba del modelo YOLO [64], ejecutado en el entorno Google Colab mediante la librería *Ultralytics* (véase Figura A3.1).

El objetivo de este apartado es proporcionar una visión detallada del procedimiento técnico seguido durante la implementación práctica del modelo, describiendo de forma estructurada cada una de las fases del proceso, además, cada bloque de código se acompaña de una explicación técnica que permite comprender la función que desempeña dentro del flujo completo del sistema.

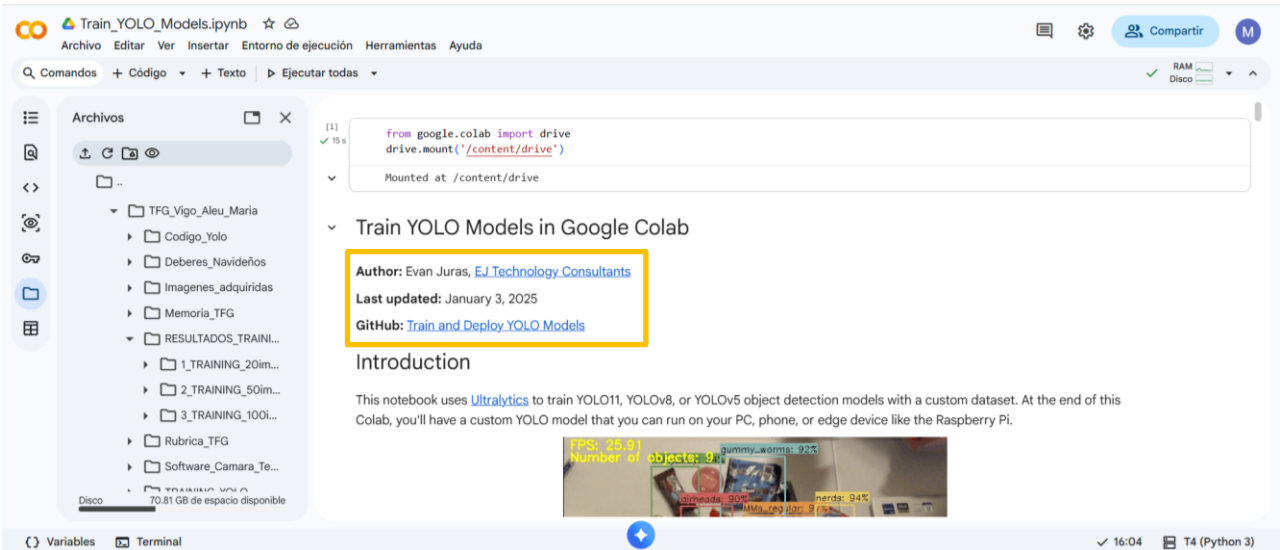


Figura A3.1 Entorno de ejecución del código YOLO en Google Colab (Elaboración propia)

PASO 1: en primer lugar, se procede al montaje de Google Drive dentro del entorno de Google Colab. Este paso permite acceder a los archivos almacenados en la nube, como el conjunto de datos, las anotaciones y archivos auxiliares necesarios para el entrenamiento del modelo. Seguidamente, se verifica la disponibilidad y utilización de la GPU, necesaria para ejecutar todo el código (puede observarse en la Figura A3.2).

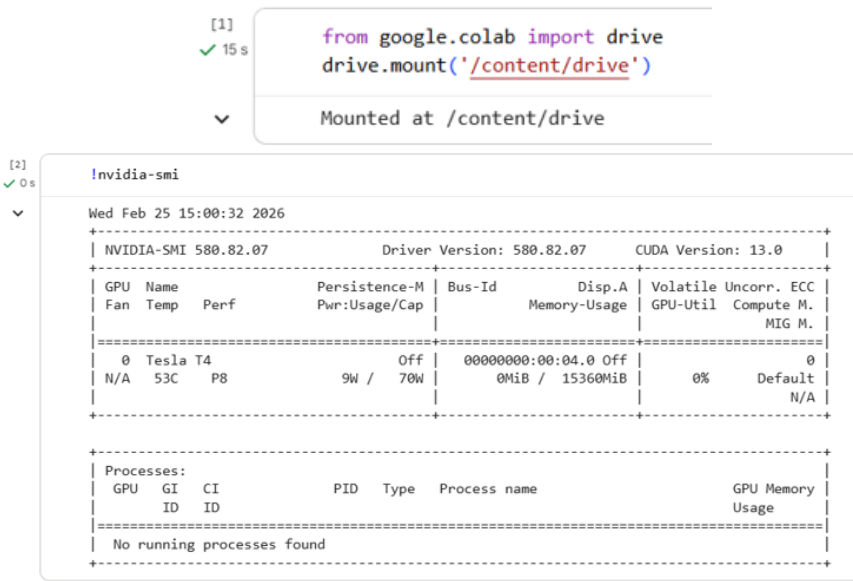


Figura A3.2 Montaje de Google Drive y comprobación del empleo de la GPU (Elaboración propia)

PASO 2: por medio de esta celda (Figura A3.3) se ejecuta una función que convierte las anotaciones de objetos creadas en VIA (almacenadas en un archivo JSON) al formato específico que requiere YOLO, extrayendo las coordenadas de cada *bounding box* y el nombre de la clase para guardarlas en un nuevo archivo .txt dentro del directorio de salida, uno por cada imagen. Para ello, se introducen las rutas hacia la ubicación de los archivos necesarios: archivo JSON con anotaciones, carpeta con imágenes a utilizar en el modelo, un archivo .txt con las clases de objetos y carpeta de salida para las anotaciones convertidas.

```
convert_via_to_yolo(
    via_json_path='/content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/via_project_26Feb2026_16h31m_json.json',
    images_dir='/content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/images',
    classes_file_path='/content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/classes.txt',
    output_dir='/content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/labels'
```

Figura A3.3 Función de llamada para conversión a formato específico YOLO (Elaboración propia)

PASO 3: en esta fase (véase Figura A3.4) se realizarán, en primer lugar, la copia del dataset por medio de una carpeta comprimida que incluye los archivos *classes*, *images* y *labels*; y en segundo lugar, la extracción de los archivos contenidos en una nueva carpeta *custom_data* para estructurar correctamente los datos conforme a los requisitos del modelo.

```
[4]
✓ 2s      from google.colab import drive
          drive.mount('/content/drive')

          !cp /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/2_TRAINING_50imag/data.zip /content

[5]
✓ 0s      # Unzip images to a custom data folder
          !unzip -q /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/2_TRAINING_50imag/data.zip -d /content/custom_data
```

Figura A3.4 Creación del dataset y extracción de archivos comprimidos (Elaboración propia)

PASO 4: para la preparación de los datos, este script está diseñado para dividir el dataset en los conjuntos de entrenamiento y validación. Como se indica en la Figura A3.5, el 80% de las imágenes y sus etiquetas van destinadas al entrenamiento, y el 20% restante a la validación.

```
[6]
✓ 0s      !wget -O /content/train_val_split.py https://raw.githubusercontent.com/EdgeElectronics/Train-and-Deploy-YOLO-Models/refs/heads/main/utis/train_val_split

          # TO DO: Improve robustness of train_val_split.py script so it can handle nested data folders, etc
          !python train_val_split.py --datapath="/content/custom_data" --train pct=0.8
```

Number of image files: 100
 Number of annotation files: 100
 Images moving to train: 80
 Images moving to validation: 20

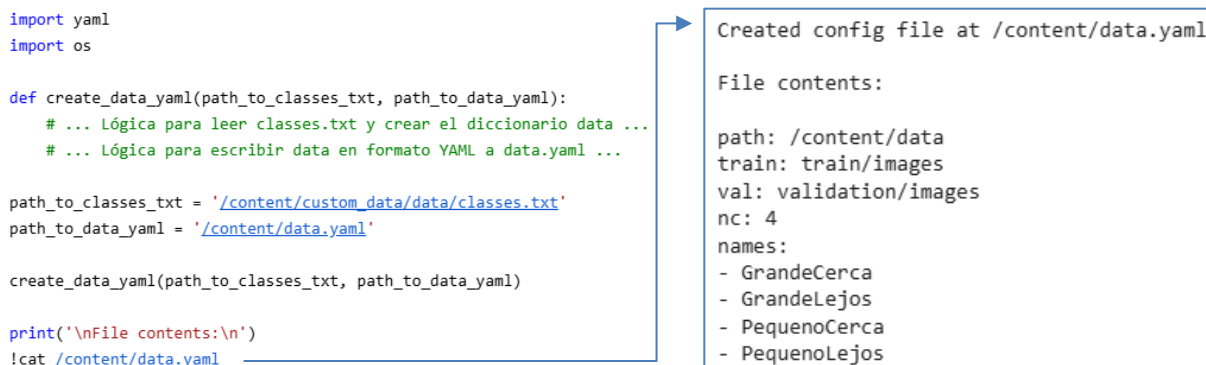
Figura A3.5 Código ejecutado para la división automática en entrenamiento y validación (Elaboración propia)

PASO 5: este comando instala la librería *Ultralytics* en el entorno de Colab (véase Figura A3.6) la cual proporciona todas las funciones necesarias para cargar modelos pre-entrenados, configurarlos y ejecutar el proceso de entrenamiento.

```
!pip install ultralytics
```

Figura A3.6 Comando de instalación de la librería *Ultralytics* (Elaboración propia)

PASO 6: este paso se encarga de generar un archivo *.yaml* que actúa como configuración del dataset. En él se especifican: ruta del conjunto de entrenamiento, ruta del conjunto de evaluación, número total de clases y nombre de cada clase (Figura A3.7).



```
import yaml
import os

def create_data_yaml(path_to_classes_txt, path_to_data_yaml):
    # ... Lógica para leer classes.txt y crear el diccionario data ...
    # ... Lógica para escribir data en formato YAML a data.yaml ...

path_to_classes_txt = '/content/custom_data/data/classes.txt'
path_to_data_yaml = '/content/data.yaml'

create_data_yaml(path_to_classes_txt, path_to_data_yaml)

print('\nFile contents:\n')
!cat /content/data.yaml
```

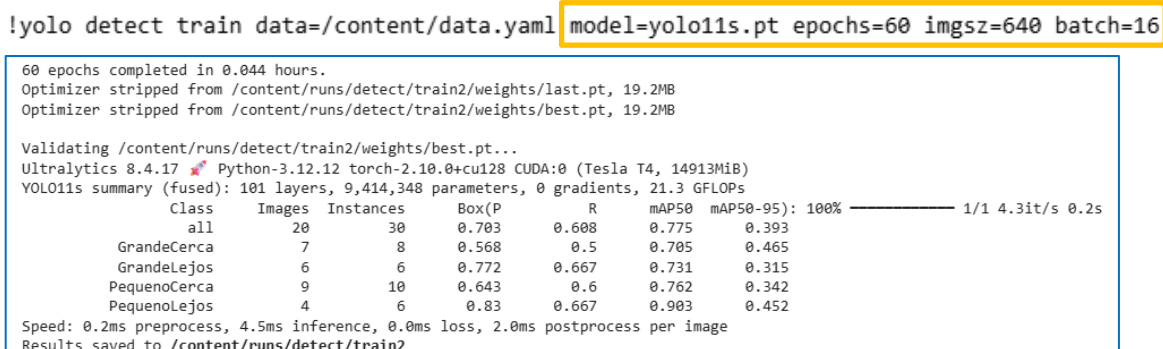
Created config file at /content/data.yaml

File contents:

```
path: /content/data
train: train/images
val: validation/images
nc: 4
names:
- GrandeCerca
- GrandeLejos
- PequenoCerca
- PequenoLejos
```

Figura A3.7 Comando y resultado de la creación del archivo .yaml (Elaboración propia)

PASO 7: este comando inicia el entrenamiento del modelo YOLO (véase Figura A3.8). En primer lugar, se selecciona el modelo base pre-entrenado *yolo11s.pt* o *yolo26s.pt*, que permite aplicar transferencia de aprendizaje (*transfer learning*). Además, se incluyen los hiperparámetros que determinan la duración del entrenamiento, la resolución a la que se procesan las imágenes y el tamaño del lote de imágenes que se procesan simultáneamente en cada paso del entrenamiento. Durante esta fase el modelo procesa las imágenes, calcula las pérdidas de localización, clasificación y confianza, ajusta los pesos mediante retropropagación y evalúa automáticamente el rendimiento en el conjunto de validación al final de cada *epoch*, permitiendo así analizar la convergencia del modelo.



```
!yolo detect train data=/content/data.yaml model=yolo11s.pt epochs=60 imgsz=640 batch=16
```

60 epochs completed in 0.044 hours.
 Optimizer stripped from /content/runs/detect/train2/weights/last.pt, 19.2MB
 Optimizer stripped from /content/runs/detect/train2/weights/best.pt, 19.2MB

Validating /content/runs/detect/train2/weights/best.pt...

Ultralytics 8.4.17 Python-3.12.12 torch-2.10.0+cu128 CUDA:0 (Tesla T4, 14913MiB)
 YOLO11s summary (fused): 101 layers, 9,414,348 parameters, 0 gradients, 21.3 GFLOPs

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95)
all	20	30	0.703	0.608	0.775	0.393
GrandeCerca	7	8	0.568	0.5	0.705	0.465
GrandeLejos	6	6	0.772	0.667	0.731	0.315
PequenoCerca	9	10	0.643	0.6	0.762	0.342
PequenoLejos	4	6	0.83	0.667	0.903	0.452

Speed: 0.2ms preprocess, 4.5ms inference, 0.0ms loss, 2.0ms postprocess per image
 Results saved to /content/runs/detect/train2

Figura A3.8 Entrenamiento y resultados del proceso de entrenamiento (Elaboración propia)

PASO 8: en esta última etapa se emplea el modelo tras el aprendizaje (*last.pt*) para ejecutar la fase de prueba (véase Figura A3.9), la cual realiza predicciones sobre nuevas imágenes no empleadas con anterioridad. Aquí se generan las *bounding boxes*, se asignan puntuaciones de confianza y se aplica el algoritmo *Non-Maximum Suppression (NMS)* para eliminar detecciones redundantes. El resultado final son las imágenes con los objetos detectados y clasificados, listas para su análisis y evaluación comparativa.

DETECCIÓN AUTOMATIZADA DE OBJETOS EN ENTORNOS MARINOS MEDIANTE VISIÓN TERMOGRÁFICA Y TÉCNICAS DE DEEP LEARNING

```
!yolo detect predict model-runs/detect/train2/weights last.pt source=/content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test save=True

Ultralytics 8.4.17 Python-3.12.12 torch-2.10.0+cu128 CUDA:0 (Tesla T4, 14913MiB)
YOLO11s summary (fused): 181 layers, 9,414,348 parameters, 0 gradients, 21.3 GFLOPs

image 1/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/E85_IR_24-01-24_0854.jpg: 480x640 (no detections), 58.7ms
image 2/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/E85_IR_24-01-24_0854_2.jpg: 480x640 1 PequenoCerca, 12.0ms
image 3/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/E85_IR_24-01-24_0905.jpg: 480x640 1 PequenoCerca, 12.0ms
image 4/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/E85_IR_24-01-24_0906.jpg: 480x640 (no detections), 12.0ms
image 5/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/E85_IR_24-01-24_0910.jpg: 480x640 1 GrandeLejos, 12.2ms
image 6/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/E85_IR_24-01-24_0911.jpg: 480x640 1 PequenoCerca, 12.2ms
image 7/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/E85_IR_29-02-24_0912.jpg: 480x640 (no detections), 12.2ms
image 8/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/E85_IR_30-01-24_0837.jpg: 480x640 (no detections), 12.2ms
image 9/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/E85_IR_30-01-24_0844.jpg: 480x640 1 PequenoCerca, 12.2ms
image 10/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/E85_IR_30-01-24_0901.jpg: 480x640 1 PequenoCerca, 12.2ms
image 11/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/IR_E85_B1_120226_B.jpg: 480x640 2 GrandeLejos, 1 PequenoLejos, 12.2ms
image 12/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/IR_E85_B1_120226_B2.jpg: 480x640 1 GrandeLejos, 2 PequenoLejos, 12.2ms
image 13/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/IR_E85_B1_140226_T.jpg: 480x640 2 PequenoCercas, 12.2ms
image 14/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/IR_E85_B1_210126_B.jpg: 480x640 1 GrandeLejos, 12.2ms
image 15/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/IR_E85_B1_240226_B.jpg: 480x640 1 PequenoLejos, 8.6ms
image 16/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/IR_E85_B1_240226_L.jpg: 480x640 (no detections), 8.6ms
image 17/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/IR_E85_B1_240226_L1.jpg: 480x640 1 GrandeLejos, 8.6ms
image 18/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/IR_E85_B1_240226_L2.jpg: 480x640 1 GrandeLejos, 8.6ms
image 19/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/IR_E85_B1_240226_L3.jpg: 480x640 1 GrandeLejos, 8.6ms
image 20/20 /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_TRAINING_YOLO/3_TRAINING_100imag/test/IR_E85_B2_140226_A.jpg: 480x640 1 PequenoLejos, 8.8ms
Speed: 1.5ms preprocess, 13.4ms inference, 1.8ms postprocess per image at shape (1, 3, 480, 640)
Results saved to /content/runs/detect/predict5

import glob
from IPython.display import Image, display
for image_path in glob.glob(f'/content/runs/detect/predict5/*.jpg')[:]:
    display(Image(filename=image_path, height=400))
    print('\n')
```

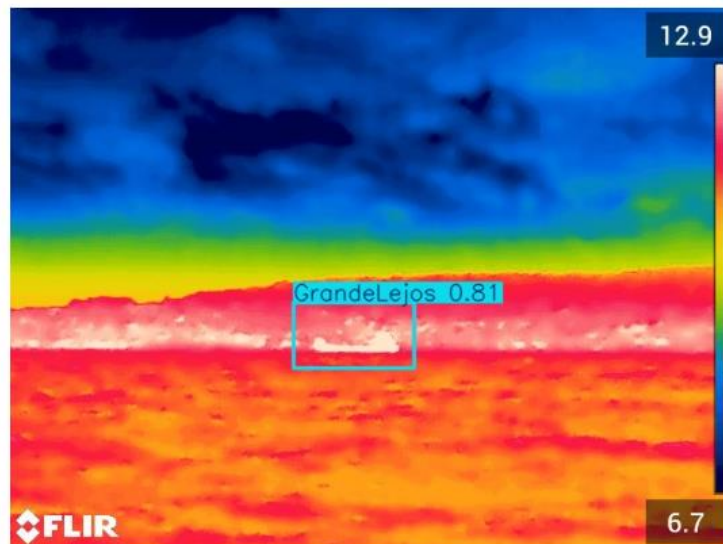


Figura A3.9 Comando para el testeo del modelo y resultado visual de las predicciones en las imágenes procesadas (Elaboración propia)

ANEXO IV: CÓDIGO MASK R-CNN

En este anexo se recogen las capturas del código utilizado para el entrenamiento, validación y testeo del modelo Mask R-CNN [65], ejecutado en el entorno Google Colab (véase Figura A4.1).

El objetivo de este apartado es proporcionar una visión detallada del procedimiento técnico seguido durante la implementación práctica del segundo modelo de detección de objetos, describiendo de forma estructurada cada una de las fases del proceso, además, cada bloque de código se acompaña de una explicación técnica que permite comprender la función que desempeña dentro del flujo completo del sistema.

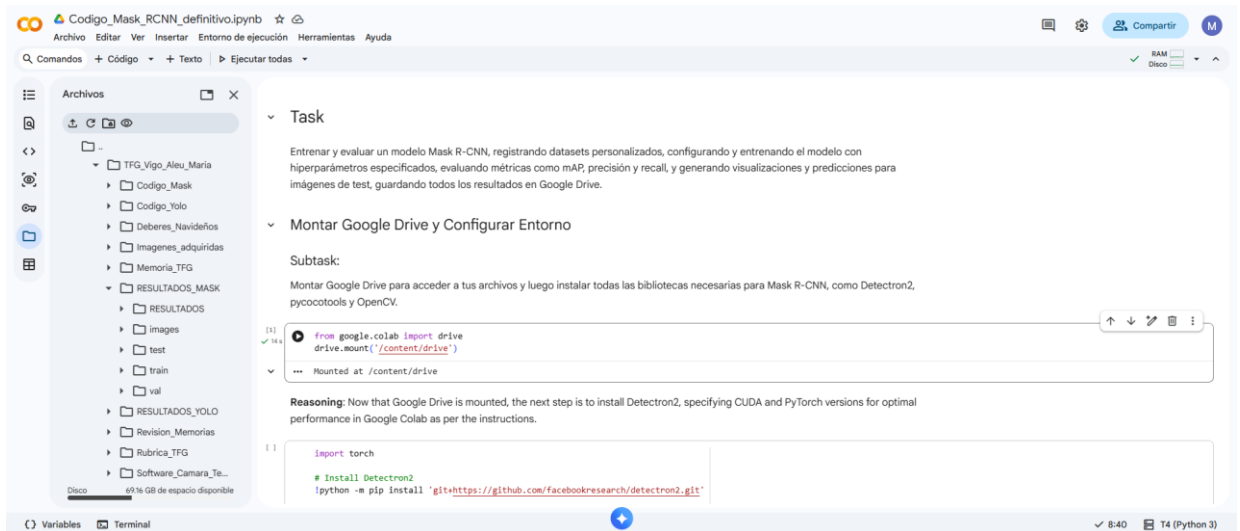


Figura A4.1 Entorno de ejecución del código Mask R-CNN en Google Colab (Elaboración propia)

PASO 1: en las primeras celdas del código (véase Figura A4.2) se ejecutan los comandos necesarios para montar Google Drive y configurar el entorno. De esta forma, el cuaderno puede acceder a los archivos y directorios almacenados en Drive. Además, se instalan las bibliotecas necesarias (*Detectron2*, *pycocotools* y *OpenCV*) para poder usar el *framework* Mask R-CNN, trabajar con el formato de anotaciones *COCO* y procesar las imágenes, respectivamente.

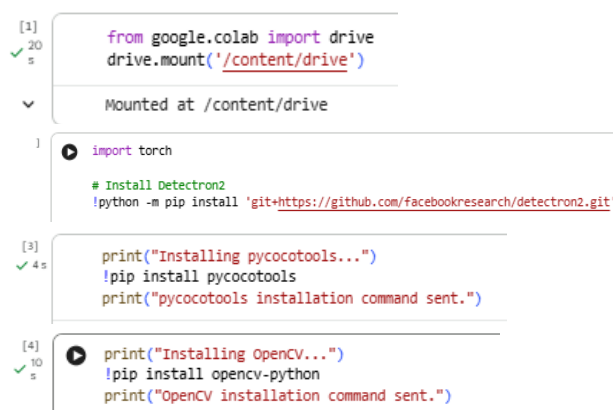


Figura A4.2 Montaje de Google Drive e instalación de librerías (Elaboración propia)

PASO 2: este paso se solicita al usuario las rutas a los directorios de imágenes y archivos JSON de anotaciones para los conjuntos de entrenamiento y validación, así como la recopilación de los hiperparámetros clave para el entrenamiento del modelo y el directorio de salida, como puede observarse en la Figura A4.3.

```
[4] print("Por favor, introduzca las rutas y los hiperparámetros:")

train_image_dir = input("1. Ruta completa al directorio de imágenes de entrenamiento (ej. '/content/drive/MyDrive/my_dataset/train_images'): ")
train_annotations_file = input("2. Ruta completa al archivo JSON de anotaciones de entrenamiento (ej. '/content/drive/MyDrive/my_dataset/train_annotations.json'): ")
val_image_dir = input("3. Ruta completa al directorio de imágenes de validación (ej. '/content/drive/MyDrive/my_dataset/val_images'): ")
val_annotations_file = input("4. Ruta completa al archivo JSON de anotaciones de validación (ej. '/content/drive/MyDrive/my_dataset/val_annotations.json'): ")

MAX_ITER = int(input("5. Número máximo de iteraciones para el entrenamiento (entero, ej. 10000): "))
img_per_batch = int(input("6. Tamaño de batch por GPU ('img_per_batch', entero, ej. 2 o 4): "))
backbone_type = input("7. Tipo de backbone para el modelo (ej. 'R_50_FPN', 'R_101_FPN'): ")
image_min_dim = int(input("8. Dimensión mínima de la imagen para el redimensionamiento ('image_min_dim', entero, ej. 800): "))
image_max_dim = int(input("9. Dimensión máxima de la imagen para el redimensionamiento ('image_max_dim', entero, ej. 1333): "))
learning_rate = float(input("10. Tasa de aprendizaje ('learning_rate', flotante, ej. 0.00025): "))
momentum = float(input("11. Valor de momentum para el optimizador (flotante, ej. 0.9): "))
weight_decay = float(input("12. Valor de weight decay (flotante, ej. 0.0001): "))
training_batch_size = int(input("13. Tamaño total de batch de entrenamiento ('training_batch_size', entero): "))
validation_batch_size = int(input("14. Tamaño de batch de validación ('validation_batch_size', entero): "))
output_dir = input("15. Ruta completa al directorio de salida para guardar los resultados (ej. '/content/drive/MyDrive/detectron2_output'): ")

print("\nConfiguraciones capturadas correctamente.")
```

```
[7] Por favor, introduzca las rutas y los hiperparámetros:
1. Ruta completa al directorio de imágenes de entrenamiento (ej. '/content/drive/MyDrive/my_dataset/train_images'): /content/drive/MyDrive/TF6_Vigo_Aleu_Maria/RESULTADOS_MASK/train
2. Ruta completa al archivo JSON de anotaciones de entrenamiento (ej. '/content/drive/MyDrive/my_dataset/train_annotations.json'): /content/drive/MyDrive/TF6_Vigo_Aleu_Maria/RESULTADOS_MASK/train/via_annotations_train_11Mar2026_15h29m_json.json
3. Ruta completa al directorio de imágenes de validación (ej. '/content/drive/MyDrive/my_dataset/val_images'): /content/drive/MyDrive/TF6_Vigo_Aleu_Maria/RESULTADOS_MASK/val
4. Ruta completa al archivo JSON de anotaciones de validación (ej. '/content/drive/MyDrive/my_dataset/val_annotations.json'): /content/drive/MyDrive/TF6_Vigo_Aleu_Maria/RESULTADOS_MASK/val/via_annotations_val_11Mar2026_15h58m_json.json
5. Número máximo de iteraciones para el entrenamiento (entero, ej. 10000): 1000
6. Tamaño de batch por GPU ('img_per_batch', entero, ej. 2 o 4): 2
7. Tipo de backbone para el modelo (ej. 'R_50_FPN', 'R_101_FPN'): R_50_FPN
8. Dimensión mínima de la imagen para el redimensionamiento ('image_min_dim', entero, ej. 800): 512
9. Dimensión máxima de la imagen para el redimensionamiento ('image_max_dim', entero, ej. 1333): 512
10. Tasa de aprendizaje ('learning_rate', flotante, ej. 0.00025): 0.001
11. Valor de momentum para el optimizador (flotante, ej. 0.9): 0.9
12. Valor de weight decay (flotante, ej. 0.0001): 0.0001
13. Tamaño total de batch de entrenamiento ('training_batch_size', entero): 16
14. Tamaño de batch de validación ('validation_batch_size', entero): 16
15. Ruta completa al directorio de salida para guardar los resultados (ej. '/content/drive/MyDrive/detectron2_output'): /content/drive/MyDrive/TF6_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS

Configuraciones capturadas correctamente.
```

Figura A4.3 Recopilación de rutas e hiperparámetros para la configuración del modelo (Elaboración propia)

PASO 3: en las siguientes celdas (véase Figura A4.4) se lleva a cabo el procesamiento de las anotaciones y el registro del *dataset*. Se cargan los archivos JSON de anotaciones de entrenamiento para extraer los nombres únicos de las clases. Luego, se define una función personalizada (*get_via_dicts*) para convertir las anotaciones de formato VIA a un formato compatible con *Detectron2*, y se registran los *datasets* de entrenamiento y validación con *Detectron2*, actualizando sus metadatos con los nombres de las clases.

```
[7] import json
from detectron2.data import DatasetCatalog, MetadataCatalog
from detectron2.data.datasets import register_coco_instances
from detectron2.data.datasets.coco import convert_to_coco_json # Import for COCO conversion

print("Detectron2 dataset registration libraries and json imported.")

... Detectron2 dataset registration libraries and json imported.
```

```
[8] with open(train_annotations_file, 'r') as f:
    train_data = json.load(f)

    class_names = set()
    for img_id, img_info in train_data.items():
        for region in img_info['regions']:
            if 'region_attributes' in region and 'Objeto' in region['region_attributes']:
                class_names.add(region['region_attributes']['Objeto'])

    class_names = sorted(list(class_names))

    print(f"Class names extracted: {class_names}")
    print(f"Number of classes: {len(class_names)}")

... Class names extracted: ['GrandeCerca', 'GrandeLejos', 'PequenoCerca', 'Pequenolejos']
Number of classes: 4
```

```
[1] register_coco_instances("my_dataset_train", {}, train_annotations_file, train_image_dir)
register_coco_instances("my_dataset_val", {}, val_annotations_file, val_image_dir)

print("Train and validation datasets registered with Detectron2.")

... Train and validation datasets registered with Detectron2.
```

```
[10] MetadataCatalog.get("my_dataset_train").set(thing_classes=class_names)
MetadataCatalog.get("my_dataset_val").set(thing_classes=class_names)

print("MetadataCatalog updated with class names for train and validation datasets.")

... MetadataCatalog updated with class names for train and validation datasets.
```

Figura A4.4 Procesamiento de anotaciones y registro de *datasets* (Elaboración propia)

PASO 4: se inicia la configuración del modelo a partir de un modelo pre-entrenado de *Detectron2*. Se ajustan los parámetros como el número de clases, configuración del optimizador (tasa de aprendizaje, iteraciones máximas, *momentum*, *weight decay*) y las dimensiones de las imágenes, utilizando los valores proporcionados en el **PASO 2**. Todo ello prepara el modelo para el inicio del entrenamiento, como puede observarse en la Figura A4.5.

```
[11]
✓ 0 s
import os
from detectron2.config import get_cfg
from detectron2 import model_zoo

print("Detectron2 configuration and model zoo libraries imported.")

... Detectron2 configuration and model zoo libraries imported.

[12]
✓ 0 s
def get_via_dicts(img_dir, annotations_file, class_names):
    with open(annotations_file, 'r') as f:
        via_data = json.load(f)

    dataset_dicts = []
    for idx, (img_id, img_info) in enumerate(via_data.items()):
        record = {}

        filename = os.path.join(img_dir, img_info['filename'])

        # Check if the file exists before attempting to read it
        if not os.path.exists(filename):
            print(f"Warning: image file not found: {filename}. Skipping.")
            continue

        height, width = cv2.imread(filename).shape[:2]

        record["file_name"] = filename
        record["height"] = height
        record["width"] = width
        record["image_id"] = idx

        annos = img_info['regions']
        objs = []
        for anno in annos:
            if not anno:
                continue
            px = anno['shape_attributes']['all_points_x']
            py = anno['shape_attributes']['all_points_y']
            poly = [(x + 0.5, y + 0.5) for x, y in zip(px, py)]
            poly = [p for x in poly for p in x] # Flatten list

            obj_class = anno['region_attributes'].get('Objeto')
            if obj_class is None or obj_class not in class_names:
                # Skip if class is not found or is None
                continue
            class_id = class_names.index(obj_class)

            obj = {
                "bbox": [min(px), min(py), max(px), max(py)], #xmin, ymin, xmax, ymax
                "bbox_mode": BoxMode.XYXY_ABS,
                "segmentation": [poly],
                "category_id": class_id,
                "iscrowd": 0
            }
            objs.append(obj)
        record["annotations"] = objs
        dataset_dicts.append(record)
    return dataset_dicts
```

Figura A4.5 Configuración inicial del modelo Mask R-CNN (Elaboración propia)

PASO 5: para el entrenamiento del modelo, se implementa una clase *CustomTrainer* que hereda de *DefaultTrainer* (entrenador estándar) para permitir evaluaciones periódicas durante el entrenamiento (véase Figura A4.6). Además, se incluyen librerías adicionales para personalizar el entrenador: *DetectionCheckpointner* para cargar y guardar *checkpoints*, *hooks* para añadir funcionalidades al ciclo de entrenamiento, *COCOEvaluator* para la evaluación del rendimiento, *DatasetMapper* y *build_detection_test_loader* para preparar los datos de validación, como puede observarse en la Figura A4.7. Este sistema sobrescribe el método *build_evaluator* y el método *build_hooks* para incluir un *EvalHook* que realiza la evaluación en el conjunto de validación cada 10 iteraciones (*cfg.TEST.EVAL_PERIOD*). Finalmente, se inicia el proceso de entrenamiento con este entrenador personalizado, guardando las métricas en *metrics.json* (véase Figura A4.8 y Figura A4.9).

```
[13]
✓ 0 s
from detectron2.engine import DefaultTrainer
from detectron2.utils.logger import setup_logger

print("Detectron2 DefaultTrainer and logger imported.")

... Detectron2 DefaultTrainer and logger imported.
```

Figura A4.6 Implementación de *DefaultTrainer* para el entrenamiento del modelo (Elaboración propia)

```
[14] import json
import os
import cv2
from detectron2.data import DatasetCatalog, MetadataCatalog
from detectron2.structures import BoxMode # Import BoxMode for bounding box format

print("Detectron2 dataset registration libraries, json, os, cv2, and BoxMode imported.")

... Detectron2 dataset registration libraries, json, os, cv2, and BoxMode imported.

[15] if "my_dataset_train" in DatasetCatalog.list():
    DatasetCatalog.remove("my_dataset_train")
if "my_dataset_val" in DatasetCatalog.list():
    DatasetCatalog.remove("my_dataset_val")

DatasetCatalog.register("my_dataset_train", lambda: get_via_dicts(train_image_dir, train_annotations_file, class_names))
DatasetCatalog.register("my_dataset_val", lambda: get_via_dicts(val_image_dir, val_annotations_file, class_names))

print("Train and validation datasets re-registered with Detectron2 using custom VIA parser.")

... Train and validation datasets re-registered with Detectron2 using custom VIA parser.

[16] import json
import os
# DatasetCatalog, MetadataCatalog already imported
from detectron2.data.datasets.coco import convert_to_coco_dict
from detectron2.data.datasets import register_coco_instances

# --- Step 1: Define output path for COCO validation annotations ---
inference_output_sub_dir = os.path.join(output_dir, 'inference')
os.makedirs(inference_output_sub_dir, exist_ok=True)
coco_val_annotations_path = os.path.join(inference_output_sub_dir, 'coco_val_annotations.json')
print(f"COCO validation annotations path defined: {coco_val_annotations_path}")

# --- Step 2: Generate COCO format annotations for validation set manually ---
print("Generating COCO format annotations for validation set manually...")

# Retrieve the dataset dictionaries using the previously registered lambda function
val_dataset_dicts = DatasetCatalog.get("my_dataset_val")

# Temporarily register val_dataset_dicts under a new name for COCO conversion
temp_dataset_name = "_temp_val_coco_conversion"

# Unregister if previously registered (for re-running this cell)
if temp_dataset_name in DatasetCatalog.list():
    DatasetCatalog.remove(temp_dataset_name)
if temp_dataset_name in MetadataCatalog.list():
    MetadataCatalog.remove(temp_dataset_name)

DatasetCatalog.register(temp_dataset_name, lambda: val_dataset_dicts)
MetadataCatalog.get(temp_dataset_name).set(thing_classes=class_names)

coco_data_dict = convert_to_coco_dict(temp_dataset_name)

with open(coco_val_annotations_path, "w") as f:
    json.dump(coco_data_dict, f, indent=4)

DatasetCatalog.remove(temp_dataset_name)
MetadataCatalog.remove(temp_dataset_name)

if os.path.exists(coco_val_annotations_path):
    print(f"COCO validation annotations successfully generated and saved to: {coco_val_annotations_path}")
else:
    print(f"Error: COCO annotations file was not found at expected path: {coco_val_annotations_path}")
print("Temporary dataset for COCO conversion unregistered.")

# --- Step 3: Register the COCO-formatted validation dataset for evaluation ---
new_val_dataset_name = "my_dataset_val_coco_eval"

# Unregister if previously registered
if new_val_dataset_name in DatasetCatalog.list():
    DatasetCatalog.remove(new_val_dataset_name)
if new_val_dataset_name in MetadataCatalog.list():
    MetadataCatalog.remove(new_val_dataset_name)

register_coco_instances(new_val_dataset_name, {}, coco_val_annotations_path, val_image_dir)
MetadataCatalog.get(new_val_dataset_name).set(thing_classes=class_names)

print(f"Validation dataset '{new_val_dataset_name}' re-registered with Detectron2 using COCO format for evaluation.")

... COCO validation annotations path defined: /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/inference/coco_val_annotations.json
Generating COCO format annotations for validation set manually...
COCO validation annotations successfully generated and saved to: /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/inference/coco_val_annotations.json
Temporary dataset for COCO conversion unregistered.
Validation dataset 'my_dataset_val_coco_eval' re-registered with Detectron2 using COCO format for evaluation.
```

**Figura A4.7 Librerías de personalización del entrenador y preparación del *dataset* para el aprendizaje
(Elaboración propia)**

```
[17]
✓ 0 s
cfig = get_cfg()

# 2. Define output directory and create it
# output_dir is already defined by user interaction.
os.makedirs(output_dir, exist_ok=True)

# 3. Select base model configuration file
config_file_path = f"COCO-InstanceSegmentation/mask_rcnn_{backbone_type}_3x.yaml"
cfg.merge_from_file(model_zoo.get_config_file(config_file_path))

# 5. Set pre-trained model weights
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url(config_file_path)

# 6. Configure number of object classes
cfg.MODEL.ROI_HEADS.NUM_CLASSES = len(class_names)

# 7. Set batch size per GPU and optimizer hyperparameters
cfg.SOLVER.IMS_PER_BATCH = ims_per_batch
cfg.SOLVER.BASE_LR = learning_rate
cfg.SOLVER.MAX_ITER = MAX_ITER
cfg.SOLVER.MOMENTUM = momentum
cfg.SOLVER.WEIGHT_DECAY = weight_decay

# 8. Configure image resizing dimensions for training and testing
cfg.INPUT.MIN_SIZE_TRAIN = image_min_dim
cfg.INPUT.MAX_SIZE_TRAIN = image_max_dim
cfg.INPUT.MIN_SIZE_TEST = image_min_dim
cfg.INPUT.MAX_SIZE_TEST = image_max_dim

# 9. Assign registered training and COCO-formatted validation datasets
cfg.DATASETS.TRAIN = ("my_dataset_train",)
cfg.DATASETS.TEST = (new_val_dataset_name,) # Use the COCO-formatted dataset for evaluation during training

# 10. Set number of workers for data loader
cfg.DATALOADER.NUM_WORKERS = 2

# 11. Assign output directory
cfg.OUTPUT_DIR = output_dir

print("Mask R-CNN model configured successfully.")

... Mask R-CNN model configured successfully.
```

```
[18]
✓ 0 s
from detectron2.checkpoint import DetectionCheckpointer
from detectron2.engine import hooks
from detectron2.evaluation import COCOEvaluator
from detectron2.data import DatasetMapper, build_detection_test_loader
from collections import OrderedDict

print("Required libraries imported for CustomTrainer.")

... Required libraries imported for CustomTrainer.
```

Figura A4.8 Configuración final del entrenamiento (Elaboración propia)

```
[ ]
cfg.TEST.EVAL_PERIOD = 10 # Evaluate every 10 iterations
print(f"cfg.TEST.EVAL_PERIOD set to: {cfg.TEST.EVAL_PERIOD}")

... cfg.TEST.EVAL_PERIOD set to: 10
```

```
[20]
✓ 0 s
from detectron2.engine import DefaultTrainer
from detectron2.engine import hooks # Import hooks explicitly

class CustomTrainer(DefaultTrainer):
    @classmethod
    def build_evaluator(cls, cfg, dataset_name, output_folder=None):
        if output_folder is None:
            output_folder = os.path.join(cfg.OUTPUT_DIR, "inference")
        return COCOEvaluator(dataset_name, output_dir=output_folder, tasks=("bbox", "segm"))

    def build_hooks(self):
        hooks_list = super().build_hooks() # Rename variable to avoid conflict
        hooks_list.append(
            hooks.EvalHook(
                self.cfg.TEST.EVAL_PERIOD, # Use self.cfg to access config within CustomTrainer
                lambda: self.test(self.cfg, self.model, self.build_evaluator(self.cfg, self.cfg.DATASETS.TEST[0]))
            )
        )
        return hooks_list

print("CustomTrainer class defined with overridden build_evaluator and build_hooks.")

... CustomTrainer class defined with overridden build_evaluator and build_hooks.
```

```
[21]
✓ 10 min
setup_logger()

trainer = CustomTrainer(cfg)
trainer.resume_or_load(resume=False) # Start training from scratch
trainer.train()

print("Mask R-CNN training process initiated with CustomTrainer and evaluation hooks.")
```

Figura A4.9 Establecimiento de las iteraciones de evaluación, inclusión de *hooks* e inicio del entrenamiento (Elaboración propia)

PASO 6: tras el entrenamiento, se importan las librerías necesarias para la validación y se reajusta el umbral de confianza (*SCORE_THRESH_TEST*) (véase Figura A4.10) del modelo para asegurar que se generaran suficientes predicciones incluso con baja confianza, lo cual es necesario para construir una matriz de confusión completa y evitar errores de *'No predictions from the model!'* en etapas posteriores. Se realiza una evaluación exhaustiva en el conjunto de validación utilizando *COCOEvaluator*, calculando métricas como *mAP*, *mAP50*, *mAP75*, *precision* y *recall* para las detecciones de cajas delimitadoras (*bbox*) y segmentación (*segm*).

```
[22]
✓ 0 s
import os
import json
import cv2
from detectron2.data import DatasetCatalog, MetadataCatalog, build_detection_test_loader
from detectron2.engine import DefaultPredictor
from detectron2.evaluation import COCOEvaluator, inference_on_dataset
# The 'L' logger was removed in newer Detectron2 versions. Using standard logger or print.
# from detectron2.utils.logger import L as logger
from detectron2.data.datasets import convert_to_coco_json
from detectron2.structures import BoxMode # Already imported, but good to ensure

print("Evaluation libraries imported.")

... Evaluation libraries imported.

[23]
✓ 1 s
cfg.MODEL_WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.05 # Lower the threshold significantly to get more predictions
predictor = DefaultPredictor(cfg)

print(f"Model weights re-loaded from: {cfg.MODEL_WEIGHTS}")
print(f"SCORE_THRESH_TEST adjusted to: {cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST}")
print("DefaultPredictor re-created with new threshold.")

... [03/11 15:49:59 d2.checkpoint.detection_checkpoint]: [DetectionCheckpoint] Loading from /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/model_final.pth
Model weights re-loaded from: /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/model_final.pth
SCORE_THRESH_TEST adjusted to: 0.05
DefaultPredictor re-created with new threshold.

[24]
✓ 1 s
evaluator = COCOEvaluator(new_val_dataset_name, output_dir=cfg.OUTPUT_DIR, tasks=["bbox", "segm"])
val_loader = build_detection_test_loader(cfg, new_val_dataset_name)
results = inference_on_dataset(predictor.model, val_loader, evaluator)

print("Evaluation results:")
```

Figura A4.10 Librerías de validación, reajuste del umbral de confianza y ejecución de la evaluación final del modelo en el conjunto de validación (Elaboración propia)

PASO 7: en esta etapa se procesa el archivo *metrics.json* para extraer las pérdidas por iteración del entrenamiento y las métricas de evaluación del conjunto de validación, preparando los datos para generar y guardar las gráficas de las curvas correspondientes como archivos PNG. Además, se calculan y grafican las matrices de confusión (normal y normalizada) para comprender el rendimiento de clasificación por clases, también en archivos PNG. Para ello el código itera sobre cada imagen en el *dataset* de validación, obtiene las predicciones del modelo (*predictor*), y compara las cajas delimitadoras y etiquetas predichas con las *ground truth*, utilizando el umbral *IoU* de 0.5 para determinar si una predicción coincide con un objeto real. Clasifica las detecciones como Verdaderos Positivos, Falsos Positivos o Falsos Negativos, y rellena las listas *y_true_cm* y *y_pred_cm* con las etiquetas correspondientes. Este proceso puede observarse en la Figura A4.11.

```

[26]
✓ 0 s
import json
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Detectron2 specific imports for confusion matrix
from detectron2.structures import Boxes
from detectron2.structures.bboxes import pairwise_iou # Corrected import

print("Libraries imported for loss curve plotting and confusion matrix generation.")

... Libraries imported for loss curve plotting and confusion matrix generation.

[26]
✓ 0 s
log_file_path = os.path.join(cfg.OUTPUT_DIR, "metrics.json")

iterations = []
total_losses = []
loss_cls = []
loss_box_regs = []
loss_masks = []
loss_rpn_cls = []
loss_rpn_locs = []

if os.path.exists(log_file_path):
    with open(log_file_path, 'r') as f:
        for line in f:
            try:
                log_entry = json.loads(line)
                if 'total_loss' in log_entry and 'iteration' in log_entry:
                    iterations.append(log_entry['iteration'])
                    total_losses.append(log_entry['total_loss'])
                    loss_cls.append(log_entry.get('loss_cls', 0.0)) # Use .get() with default for robustness
                    loss_box_regs.append(log_entry.get('loss_box_reg', 0.0))
                    loss_masks.append(log_entry.get('loss_mask', 0.0))
                    loss_rpn_cls.append(log_entry.get('loss_rpn_cls', 0.0))
                    loss_rpn_locs.append(log_entry.get('loss_rpn_loc', 0.0))
            except json.JSONDecodeError:
                # Skip lines that are not valid JSON
                continue
        print(f"Extracted {len(iterations)} log entries from {log_file_path}")
    else:
        print(f"Warning: metrics.json not found at {log_file_path}. Cannot plot loss curves.")

... Extracted 50 log entries from /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/metrics.json

[27]
✓ 0 s
loss_types = {
    "total_loss": total_losses,
    "loss_cls": loss_cls,
    "loss_box_reg": loss_box_regs,
    "loss_mask": loss_masks,
    "loss_rpn_cls": loss_rpn_cls,
    "loss_rpn_loc": loss_rpn_locs
}

for loss_name, loss_values in loss_types.items():
    if loss_values:
        plt.figure(figsize=(10, 6))
        plt.plot(iterations, loss_values, label=loss_name.replace('_', ' ').title())
        plt.xlabel("Iteration")
        plt.ylabel("Loss")
        plt.title(f"Training Loss Curve: {loss_name.replace('_', ' ').title()}")
        plt.grid(True)
        plt.legend()
        plot_file_path = os.path.join(cfg.OUTPUT_DIR, f"{loss_name}_curve.png")
        plt.savefig(plot_file_path)
        plt.close()
        print(f"Saved {loss_name} curve to {plot_file_path}")
    else:
        print(f"No data for {loss_name} curve to plot.")

print("All loss curves generated and saved.")

... Saved total_loss curve to /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/total_loss_curve.png
... Saved loss_cls curve to /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/loss_cls_curve.png
... Saved loss_box_reg curve to /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/loss_box_reg_curve.png
... Saved loss_mask curve to /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/loss_mask_curve.png
... Saved loss_rpn_cls curve to /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/loss_rpn_cls_curve.png
... Saved loss_rpn_loc curve to /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/loss_rpn_loc_curve.png
... All loss curves generated and saved.

[28]
✓ 1 s
num_classes = len(class_names)
background_class_id = num_classes # Background/No Match class

y_true_cm = []
y_pred_cm = []

IOU_THRESHOLD = 0.5

print(f"Processing {len(val_dataset_dicts)} validation images for confusion matrix...")

[29]
✓ 0 s
display_labels = class_names + ["No Match/Background"]

# Generate Normal Confusion Matrix
plt.figure(figsize=(12, 10))
# Swapping y_true_cm and y_pred_cm here to make rows=predictions, columns=true values
cm_normal = confusion_matrix(y_pred_cm, y_true_cm, labels=list(range(num_classes + 1)))
sns.heatmap(cm_normal, annot=True, fmt='d', cmap='Blues', xticklabels=display_labels, yticklabels=display_labels)
plt.xlabel("True Label")
plt.ylabel("Predicted Label")
plt.title("Confusion Matrix (Normal)")
plot_file_path_normal = os.path.join(cfg.OUTPUT_DIR, "confusion_matrix_normal.png")
plt.savefig(plot_file_path_normal)
plt.close()
print(f"Saved Normal Confusion Matrix to {plot_file_path_normal}")

... Saved Normal Confusion Matrix to /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/confusion_matrix_normal.png

[30]
✓ 0 s
display_labels = class_names + ["No Match/Background"]

# Generate Normalized Confusion Matrix
# Handle potential division by zero for columns with no true labels
# Normalize by columns (sum of true labels) so each column sums to 1
cm_normalized = cm_normal.astype('float') / cm_normal.sum(axis=0)[np.newaxis, :]
cm_normalized = np.nan_to_num(cm_normalized, nan=0.0, posinf=0.0, neginf=0.0)

plt.figure(figsize=(12, 10))
sns.heatmap(cm_normalized, annot=True, fmt='.2f', cmap='Blues', xticklabels=display_labels, yticklabels=display_labels)
plt.xlabel("True Label")
plt.ylabel("Predicted Label")
plt.title("Confusion Matrix (Normalized)")
plot_file_path_normalized = os.path.join(cfg.OUTPUT_DIR, "confusion_matrix_normalized.png")
plt.savefig(plot_file_path_normalized)
plt.close()
print(f"Saved Normalized Confusion Matrix to {plot_file_path_normalized}")

... Saved Normalized Confusion Matrix to /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/confusion_matrix_normalized.png

```

Figura A4.11 Generación de gráficos de métricas de validación y matrices de confusión (Elaboración propia)

PASO 8: en primer lugar, extrae las métricas *mAP* (Figura A4.12) por clase (para *bbox* y *segm*) de los resultados de la evaluación (*results*). Combina estas métricas en un diccionario y las guarda en un archivo JSON (*per_class_metrics.json*) en el directorio de salida. Esto permite un análisis detallado del rendimiento por cada categoría de objeto. Seguidamente, itera a través de la selección completa de las imágenes del conjunto de validación, obteniendo predicciones del modelo y genera visualizaciones que comparan las anotaciones reales (*ground truth*) con las predicciones del modelo, guardando estas imágenes anotadas en el directorio de salida para un análisis visual, con nombres que indican si son '*GT*' o '*PRED*', como puede observarse en la Figura A4.13 y Figura A4.14.

```
[31]
✓ 0 s
import json
import os

# Extract per-class mAP (bbox)
bbox_per_class_ap = results.get('bbox', {}).get('AP-category', {})

# Extract per-class mAP (segm)
segm_per_class_ap = results.get('segm', {}).get('AP-category', {})

# Combine into a single dictionary
per_class_metrics = {
    "bbox_per_class_ap": bbox_per_class_ap,
    "segm_per_class_ap": segm_per_class_ap
}

# Save to a JSON file in the output directory
metrics_output_path = os.path.join(output_dir, "per_class_metrics.json")
with open(metrics_output_path, "w") as f:
    json.dump(per_class_metrics, f, indent=4)

print(f"Per-class mAP metrics saved to: {metrics_output_path}")

... Per-class mAP metrics saved to: /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/per_class_metrics.json
```

Figura A4.12 Extracción y guardado de métricas *mAP* de los resultados de la evaluación (Elaboración propia)

```
[32]
✓ 0 s
from detectron2.engine import DefaultPredictor
from detectron2.utils.visualizer import Visualizer
from detectron2.utils.visualizer import ColorMode

print("Detectron2 visualization libraries imported.")

... Detectron2 visualization libraries imported.

[33]
✓ 0 s
val_metadata = MetadataCatalog.get(new_val_dataset_name)
print(f"Visualizer will use metadata from: {val_metadata.name}")

... Visualizer will use metadata from: my_dataset_val_coco_eval

import random

# Select a subset of images for visualization (e.g., first 5 or a random sample)
# To visualize all images, set num_images_to_visualize to the total length of the dataset.
num_images_to_visualize = len(val_dataset_dicts)

# For consistent output, we'll just take the first 'num_images_to_visualize' images
sampled_val_dicts = val_dataset_dicts[:num_images_to_visualize]

print(f"Selected {num_images_to_visualize} images for visualization.")

... Selected 20 images for visualization.

[35]
✓ 3 s
for d in sampled_val_dicts:
    img = cv2.imread(d["file_name"])
    if img is None:
        print(f"Warning: Could not read image {d['file_name']}. Skipping visualization for this image.")
        continue
```

Figura A4.13 Obtención metadatos de validación y selección de imágenes para la preparación del *Visualizer* (Elaboración propia)

```

[36] ✓ 0 s
iterations_eval = []
bbox_AP = []
bbox_AP50 = []
bbox_AP75 = []
segm_AP = []
segm_AP50 = []
segm_AP75 = []

print("Initialized lists for evaluation metrics.")

... Initialized lists for evaluation metrics.

[ ]

import json
import os

log_file_path = os.path.join(cfg.OUTPUT_DIR, "metrics.json")

iterations = []
total_losses = []
loss_cls = []
loss_box_regs = []
loss_masks = []
loss_rpn_cls = []
loss_rpn_locs = []
validation_losses = [] # New list for validation loss

[38] ✓ 0 s
log_file_path = os.path.join(cfg.OUTPUT_DIR, "metrics.json")

if os.path.exists(log_file_path):
    with open(log_file_path, 'r') as f:
        for line in f:
            try:
                log_entry = json.loads(line)
                # Check if it's an evaluation entry by looking for 'bbox/AP' or 'segm/AP'
                if 'bbox/AP' in log_entry or 'segm/AP' in log_entry:
                    iterations_eval.append(log_entry['iteration'])

                    # Bbox metrics
                    bbox_AP.append(log_entry.get('bbox/AP', np.nan))
                    bbox_AP50.append(log_entry.get('bbox/AP50', np.nan))
                    bbox_AP75.append(log_entry.get('bbox/AP75', np.nan))

                    # Segm metrics
                    segm_AP.append(log_entry.get('segm/AP', np.nan))
                    segm_AP50.append(log_entry.get('segm/AP50', np.nan))
                    segm_AP75.append(log_entry.get('segm/AP75', np.nan))
            except json.JSONDecodeError:
                # skip lines that are not valid JSON
                continue
    print(f"Extracted {len(iterations_eval)} evaluation log entries from {log_file_path}")
else:
    print(f"Warning: metrics.json not found at {log_file_path}. Cannot extract evaluation metrics.")

... Extracted 51 evaluation log entries from /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/metrics.json

[39] ✓ 0 s
metric_types = {
    "bbox_AP": bbox_AP,
    "bbox_AP50": bbox_AP50,
    "bbox_AP75": bbox_AP75,
    "segm_AP": segm_AP,
    "segm_AP50": segm_AP50,
    "segm_AP75": segm_AP75
}

for metric_name, metric_values in metric_types.items():
    if metric_values and iterations_eval:
        plt.figure(figsize=(10, 6))
        plt.plot(iterations_eval, metric_values, label=metric_name.replace('_', ' ').title())
        plt.xlabel("Iteration")
        plt.ylabel(metric_name.replace('_', ' ').title())
        plt.title(f"Validation {metric_name.replace('_', ' ').title()} Over Iterations")
        plt.grid(True)
        plt.legend()
        plot_file_path = os.path.join(cfg.OUTPUT_DIR, f"{metric_name}_curve.png")
        plt.savefig(plot_file_path)
        plt.close()
        print(f"Saved {metric_name} curve to {plot_file_path}")
    else:
        print(f"No data for {metric_name} curve to plot.")

print("All evaluation metric curves generated and saved.")

... Saved bbox_AP curve to /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/bbox_AP_curve.png
... Saved bbox_AP50 curve to /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/bbox_AP50_curve.png
... Saved bbox_AP75 curve to /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/bbox_AP75_curve.png
... Saved segm_AP curve to /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/segm_AP_curve.png
... Saved segm_AP50 curve to /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/segm_AP50_curve.png
... Saved segm_AP75 curve to /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/segm_AP75_curve.png
... All evaluation metric curves generated and saved.

```

Figura A4.14 Carga de imágenes, iteraciones y obtención de predicciones para su visualización (Elaboración propia)

PASO 9: para la inferencia del modelo en nuevas imágenes de prueba (véase Figura A4.15), se solicita la ruta al directorio con dichas imágenes y se configura el *predictor* con el modelo final entrenado y un umbral de confianza de 0.3 para la inferencia. Finalmente, se realizan las predicciones en cada imagen de prueba y se guardan las visualizaciones de las detecciones (cajas delimitadoras y máscaras) en una subcarpeta específica dentro del directorio de imágenes de prueba, indicando el modelo utilizado.

```
[40]
✓ 4s
test_image_dir = input("Por favor, introduzca la ruta completa a la carpeta de Google Drive que contiene las imágenes de prueba para la inferencia (ej. '/content/drive/MyDrive/my_test_images'): ")
print(f"Directorio de imágenes de prueba para inferencia: {test_image_dir}")

... Por favor, introduzca la ruta completa a la carpeta de Google Drive que contiene las imágenes de prueba para la inferencia (ej. '/content/drive/MyDrive/my_test_images'): /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_YOLO/1_E85_v26/test
Directorio de imágenes de prueba para inferencia: /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_YOLO/1_E85_v26/test

[41]
✓ 0s
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.3 # You can adjust this threshold as needed

print(f"Model weights path set to: {cfg.MODEL.WEIGHTS}")
print(f"Prediction score threshold set to: {cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST}")

... Model weights path set to: /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/model_final.pth
Prediction score threshold set to: 0.3

[42]
✓ 1s
predictor = DefaultPredictor(cfg)

print("DefaultPredictor initialized with the final trained model.")

... [03/11 16:04:32 d2.checkpoint.detection_checkpoint]: [DetectionCheckpointer] Loading from /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_MASK/RESULTADOS/model_final.pth ...
DefaultPredictor initialized with the final trained model.

[43]
✓ 0s
inference_output_dir = os.path.join(test_image_dir, "inference_results_model_final")
os.makedirs(inference_output_dir, exist_ok=True)

print(f"Inference results will be saved to: {inference_output_dir}")

... Inference results will be saved to: /content/drive/MyDrive/TFG_Vigo_Aleu_Maria/RESULTADOS_YOLO/1_E85_v26/test/inference_results_model_final

[44]
✓ 21s
from detectron2.utils.visualizer import Visualizer
from detectron2.utils.visualizer import ColorMode

# Get the metadata for visualization
# We will use the metadata from the training dataset as it contains the class names
# and we are only displaying predictions here, not evaluating.
# However, it's safer to use the metadata from the evaluation dataset for consistency if it was properly registered with class names.
# Let's re-ensure val_metadata is available or re-fetch it.
val_metadata = MetadataCatalog.get("my_dataset_train") # Reusing train metadata as it has class_names

image_files = [f for f in os.listdir(test_image_dir) if os.path.isfile(os.path.join(test_image_dir, f))]

print(f"Starting inference on {len(image_files)} images in {test_image_dir}...")
```

Figura A4.15 Proceso de inferencia del modelo para imágenes de test no utilizadas (Elaboración propia)

ANEXO V: DECLARACIÓN DE USO DE INTELIGENCIA ARTIFICIAL

En la elaboración del presente Trabajo de Fin de Grado, se ha hecho uso de herramientas de Inteligencia Artificial Generativa con el propósito de mejorar la redacción, coherencia y precisión del lenguaje empleado en la documentación. En particular, la IA ha sido utilizada para reforzar la expresión escrita, elevar el registro del lenguaje técnico y garantizar la claridad en la exposición de conceptos complejos relacionados con la termografía infrarroja, la visión artificial, y las redes neuronales convolucionales. Además, su uso ha contribuido a la eficacia en la búsqueda y selección de información, facilitando la filtración y análisis de referencias bibliográficas relevantes para este estudio. Esta capacidad ha permitido una mejor organización de los contenidos y un enfoque más riguroso en el respaldo teórico del trabajo.

Es importante recalcar que el uso de IA en este trabajo ha tenido un fin estrictamente auxiliar y académico, promoviendo el aprendizaje y el desarrollo de competencias en la utilización de herramientas tecnológicas avanzadas. La inteligencia artificial se ha integrado como un recurso de apoyo en la optimización del proceso de redacción y análisis de información, sin reemplazar en ningún momento el razonamiento crítico, la interpretación de los resultados obtenidos ni las conclusiones derivadas de la investigación.

El uso de estas herramientas se ha realizado bajo una supervisión constante, verificando la veracidad de la información generada y asegurando el cumplimiento de las normativas éticas y académicas vigentes. Por tanto, el uso de inteligencia artificial se ha llevado a cabo de forma responsable y transparente, como una herramienta complementaria dentro del proceso de aprendizaje y desarrollo del Trabajo Fin de Grado.