

Trabajo Fin de Grado

Detección temprana de anomalías térmicas en escenas a larga distancia mediante una cámara térmico-óptica

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Alejandro Vialard Santana

Tutor académico: Nelson Monzón López y Agustín Salgado de la Nuez

Tutor empresa: Sergio Ballines Barrera

9 de diciembre de 2024

SOLICITUD DE DEFENSA DE TRABAJO DE FIN DE TÍTULO

D. Alejandro Vialard Santana, autor del trabajo **Detección temprana de anomalías térmicas en escenas a larga distancia mediante una cámara térmico-óptica**, correspondiente a la titulación Grado en Ingeniería Informática, en colaboración con la empresa **Qualitas Artificial Intelligence And Science S.A. (QAISC)**.

SOLICITA

que se inicie el procedimiento de defensa del mismo, para lo que se adjunta la documentación requerida, haciendo constar que

☒ se autoriza / ☐ no se autoriza la grabación en audio de la exposición y turno de preguntas.

Asimismo, con respecto al registro de la propiedad intelectual/industrial del TFT, declara que:

☐ Se ha iniciado o hay intención de iniciarlo (defensa no pública).

☒ No está previsto.

Y para que así conste firma la presente. (fecha en firma electrónica)

El estudiante

Fdo. _____

A rellenar y firmar **obligatoriamente** por el/la/los/las tutores

En relación con la presente solicitud, se informa: (firmar donde corresponda)

Positivamente (en caso de detección de copia, esta firma quedará invalidada)

Fdo. _____

Negativamente (justificación en TFT05)

Fdo. _____

DIRECTORA DE LA ESCUELA DE INGENIERÍA INFORMÁTICA

Agradecimientos

En primer lugar, quiero expresar mi más profundo agradecimiento a mi familia, especialmente a mis padres, por estar siempre a mi lado, brindándome su apoyo incondicional a lo largo de mi carrera universitaria y en el desarrollo de este proyecto.

Asimismo, quiero expresar mi gratitud a mi círculo cercano de amigos, quienes, con su apoyo incondicional y escucha atenta, son uno de los pilares fundamentales de mi vida y mi carrera. A cada uno de ellos, gracias por estar ahí en los momentos en que más los necesitaba.

A mis tutores, Nelson Monzón, Sergio Ballines y Agustín Salgado, les agradezco por proporcionarme la orientación y el apoyo necesarios para llevar a cabo este proyecto con éxito.

Finalmente, quiero expresar mi gratitud a la Plataforma Oceánica de Canarias (PLOCAN) y al equipo de Qualitas Artificial Intelligence and Science (QAISC) por ofrecerme los recursos y la oportunidad de realizar esta investigación.

Resumen

En este trabajo de fin de título (TFT) se investigaron diversas técnicas de aprendizaje automático para desarrollar un modelo capaz de detectar anomalías térmicas, como fuego y humo, en entornos portuarios, empleando una cámara Hikvision bi-espectro. Se exploraron y evaluaron distintos modelos hasta lograr un enfoque óptimo. Para el entrenamiento, se creó un dataset sintético con imágenes térmicas de incendios, que luego se combinó con imágenes reales para validar el rendimiento del sistema. El modelo final es capaz de generar alertas tempranas, contribuyendo a la prevención de daños y a la seguridad en áreas costeras. Este enfoque integra técnicas avanzadas para mejorar la vigilancia en infraestructuras portuarias.

Abstract

In this bachelor's thesis, some machine learning techniques were researched to develop a model capable of detecting thermal anomalies, such as fire and smoke, in port environments using a Hikvision bi-spectrum camera. Several models were explored and evaluated to achieve an optimal approach. A synthetic dataset with thermal images of fires was created for training, and it was combined with real images to validate the system's performance. The final model can generate early alerts, contributing to damage prevention and safety in coastal areas. This approach integrates advanced techniques to enhance surveillance in port infrastructures.

Índice general

1. Introducción	1
1.1. Objetivos del proyecto	2
1.2. Competencias Específicas	3
1.3. Alineamiento con los Objetivos de Desarrollo Sostenible	4
1.4. Presupuesto	6
2. Estado actual	8
2.1. Clasificación	9
2.2. Detección de Objetos	13
2.3. Segmentación	16
2.4. Generative Adversarial Networks (GANs) y sus aplicaciones en la Transformación de Imágenes	21
2.5. Modelos de Procesamiento de Texto a Imagen	26
3. Metodología de Trabajo	29
3.1. Metodología Basada en Objetivos	29
3.2. Implementación de Kanban con Trello	30
4. Herramientas y Tecnologías	32
4.1. Lenguajes de Programación y Bibliotecas	32
4.2. Frameworks de Aprendizaje Profundo	33
4.3. Herramientas de Desarrollo y Control de Versiones	33
4.4. Herramientas de Contenerización y Despliegue	33
4.5. Herramientas de Anotación y Generación de Datos	34
4.6. Hardware Especializado	34
5. Análisis y Desarrollo del proyecto	35
5.1. Análisis del Funcionamiento de la Cámara PTZ y Planificación	35
5.2. Investigación de Modelos y Arquitecturas para la Detección Térmica	37
5.3. Creación de Imágenes Sintéticas usando Redes Generativas Adversariales (GANs)	56
5.4. Generación de Imágenes Sintéticas mediante Modelos de Texto a Imagen . .	64
5.5. YOLO para la detección de anomalías térmicas	68
6. Conclusiones y trabajo futuro	84

Índice de figuras

2.1.	Comparación entre una neurona biológica y una neurona artificial. La imagen muestra las analogías estructurales entre ambos tipos de neuronas, destacando elementos como las dendritas (entradas), el cuerpo celular (procesamiento) y el axón (salida) en la neurona biológica, y sus equivalentes en la neurona artificial. Fuente: Adaptado de “Procedimiento para el pronóstico de la demanda mediante redes neuronales artificiales” por Lao León et al. (2017), Ciencias Holguín, 23(1), p. 5.	10
2.2.	Arquitectura de U-Net. Cada bloque azul representa un mapa de características multi-canal, con el número de canales indicado en la parte superior de cada bloque. Las dimensiones espaciales x - y se encuentran en la esquina inferior izquierda. Las cajas blancas representan mapas de características copiados mediante las conexiones de salto (<i>skip connections</i>). Las flechas denotan las diferentes operaciones, como convoluciones, <i>max pooling</i> , y convoluciones transpuestas. Fuente: Adaptado de Ronneberger et al. (2015) [42].	19
2.3.	Esquema de funcionamiento de una Red Generativa Antagónica (GAN). En este diagrama, la red generadora (en azul) produce ejemplos falsos a partir de ruido o imágenes iniciales aleatorias. Estos ejemplos generados se envían al discriminador (en rojo), el cual también recibe ejemplos reales del conjunto de datos de entrenamiento. El discriminador evalúa si cada imagen es real o generada, proporcionando retroalimentación a la red generadora para mejorar sus resultados con el tiempo. Fuente: Adaptado de IBM Developer https://developer.ibm.com/articles/generative-adversarial-networks-explained/	22
2.4.	Ejemplos del funcionamiento de CycleGAN en la traducción de imágenes sin pares de correspondencia. Cada fila muestra distintas aplicaciones de traducción de imágenes: de izquierda a derecha, se muestran la imagen de entrada x , la imagen generada $G(x)$ tras la traducción y la imagen reconstruida $F(G(x))$. Los ejemplos ilustran diversas tareas de traducción: de fotos a estilo Cezanne, de caballos a cebras, de invierno a verano en Yosemite, y de fotos aéreas a mapas de Google. La red generativa antagónica con consistencia cíclica permite transformar imágenes de un dominio a otro y volver al original manteniendo coherencia. Fuente: Adaptado de Zhu et al. (2017) [55].	23

2.5.	Proceso de entrenamiento de una GAN condicional en Pix2Pix para mapear contornos a fotografías. El generador G toma una imagen de contorno x y produce una imagen fotográfica $G(x)$. El discriminador D recibe pares de imágenes {contorno, foto} y aprende a diferenciar entre pares reales (del conjunto de entrenamiento) y pares sintéticos (creados por G). A diferencia de una GAN estándar, tanto el generador como el discriminador en Pix2Pix reciben la imagen de entrada como condicionante. Fuente: Adaptado de Isola et al. (2017) [20].	24
2.6.	Ejemplos de traducción de imágenes utilizando el modelo MUNIT. La imagen muestra transformaciones estilísticas entre diferentes clases de animales, con la imagen de entrada en la columna “Input” y las traducciones en la columna “Sample translations”. MUNIT permite realizar estas traducciones manteniendo las características de contenido pero modificando el estilo. Fuente: Adaptado de Huang et al. (2018) [18].	25
3.1.	Tablero Kanban en Trello utilizado para la gestión de tareas en el desarrollo del Trabajo Fin de Grado	31
5.1.	Imágenes capturadas por la cámara PTZ Biespectro térmico-óptica de Hikvision , mostrando comparaciones entre el sensor <i>óptico</i> (izquierda) y el <i>térmico</i> (derecha). La fila superior corresponde a la Plataforma Oceánica de Canarias (PLOCAN) y la fila inferior a un aerogenerador	36
5.2.	Comparación de imágenes RGB y térmicas en diferentes modalidades capturadas en el dataset FLAME. La imagen muestra capturas desde UAVs y drones, utilizando cámaras en formato RGB y térmico (<i>WhiteHot</i> , <i>Fusion</i> , y <i>GreenHot</i>), que permiten la detección precisa de incendios y su evolución. Las primeras filas representan imágenes ópticas, mientras que las últimas tres muestran diferentes formas de visualización térmica del fuego y el entorno circundante. Fuente: Adaptado de <i>FLAME dataset: Aerial imagery pile burn detection using drones (UAVs)</i> por Alireza Shamsoshoara et al. (2021), <i>IEEE Dataport</i>	39
5.3.	Esquema de la arquitectura del modelo Xception utilizado para la clasificación de imágenes de incendios en “Fuego” y “No Fuego”. Fuente: Adaptado del repositorio de GitHub <i>Fire Detection UAV Aerial Image Classification Segmentation</i> por Alireza Shamsoshoara et al. (2021) https://github.com/AlirezaShamsoshoara/Fire-Detection-UAV-Aerial-Image-Classification-Segmentation-UnmannedAerialVehicle	41
5.4.	Imágenes obtenidas del dataset de FLAME. La columna de la izquierda representan la clase “Fuego” y la derecha “No fuego”, siendo la fila superior en formato RGB y la inferior IR en escala <i>Whitehot</i>	42
5.5.	Gráfica de rendimiento del modelo de clasificación binaria Xception durante 40 épocas en el conjunto de datos RGB de FLAME, validando pérdida y precisión.	44
5.6.	Gráfica de rendimiento del modelo de clasificación binaria Xception durante 40 épocas en el conjunto de datos IR de FLAME, validando pérdida y precisión.	45

5.7. Matriz de confusión normalizada del modelo Xception entrenado en el conjunto RGB para la clasificación de fuego/no-fuego en el conjunto de prueba. . . .	46
5.8. Matriz de confusión normalizada del modelo Xception entrenado en el conjunto IR para la clasificación de fuego/no-fuego en el conjunto de prueba.	47
5.9. Esquema de la arquitectura del modelo U-Net utilizado para la segmentación de imágenes de incendios. Fuente: Adaptado del repositorio de GitHub <i>Fire Detection UAV Aerial Image Classification Segmentation</i> por Alireza Shamsoshoara et al. (2021) https://github.com/AlirezaShamsoshoara/Fire-Detection-UAV-Aerial-Image-Classification-Segmentation-UnmannedAerialVehicle	48
5.10. Comparación de una imagen RGB capturada por la cámara DJI Phantom 3 del dataset FLAME y su respectiva máscara binaria etiquetada como “fuego”.	50
5.11. Imagen que muestra como primera fila imágenes RGB del dataset FLAME, seguido de las máscaras <i>ground truth</i> y las máscara predichas por el modelo U-Net.	51
5.12. Resultados de las máscaras binarias predichas por el modelo de segmentación U-Net en imágenes de incendios marítimos.	52
5.13. Comparación visual entre las imágenes originales del conjunto de datos IR de FLAME y las máscaras binarias generadas. La fila superior muestra los frames originales capturados en infrarrojo, mientras que la fila inferior presenta las máscaras binarias correspondientes, creadas utilizando técnicas de procesamiento de imagen en OpenCV. Estas máscaras resaltan las áreas de interés (zonas de fuego) mediante umbrales específicos y operaciones morfológicas, facilitando la segmentación precisa de las anomalías térmicas en las imágenes IR.	53
5.14. Imagen que muestra como primera fila imágenes IR del dataset FLAME, seguido de las máscaras <i>ground truth</i> y las máscara predichas por el modelo U-Net.	55
5.15. Resultados de las máscaras binarias predichas por el modelo de segmentación U-Net en imágenes de incendios y otros entornos que no presentan fuego.	56
5.16. Resultados del primer intento de transformación de dominios RGB a imágenes térmicas mediante CycleGAN, siendo la fila superior las imágenes RGB reales y la fila inferior la transformación generada.	59
5.17. Imágenes RGB asociadas a su respectivo frame en IR obtenidas mediante la cámara PTZ proporcionada por QAISC y PLOCAN. Las diferencias mínimas entre imágenes del mismo frame se deben a variaciones en las lentes óptica y térmica, como el tamaño del sensor o la distorsión.	61
5.18. Resultados de transformación de dominios RGB a imágenes térmicas mediante Pix2Pix, siendo la fila superior las imágenes RGB reales y la fila inferior la transformación generada.	61
5.19. Resultados de transformación de dominios RGB a imágenes térmicas mediante MUNIT, siendo la primera fila las imágenes originales RGB y cada una de las otras filas un resultado distinto generado en una misma inferencia.	63
5.20. Resultados obtenidos con el modelo FLUX.1 [schnell] utilizando el prompt anterior para generar imágenes IR sintéticas de incendios en barcos.	65

5.21. Resultados obtenidos con el modelo Juggernaut XL X utilizando el prompt anterior para generar imágenes IR sintéticas de incendios en barcos.	66
5.22. Imagen que simula la escala <i>WhiteHot</i> mediante el modelo de Juggernaut XL X en base al estilo de una imagen de referencia.	67
5.23. Ejemplo de imágenes finales generadas por Juggernaut XL X a partir de algunos de los anteriores prompts.	68
5.24. Interfaz web de CVAT para crear las cajas delimitadores de las clases de fuego (naranja) y humo (verde) del conjunto de datos.	70
5.25. Ejemplos de imágenes de incendios del conjunto de datos CORSICAN. En la primera fila se presentan las imágenes en formato RGB, en la segunda fila se muestran sus correspondientes versiones en infrarrojo cercano (NIR), y en la tercera fila se encuentran las máscaras binarias que identifican las áreas afectadas por el fuego. Estas tres representaciones permiten una evaluación completa de las condiciones visuales y térmicas de los incendios.	72
5.26. Comparación de los resultados de entrenamiento de los modelos YOLOv8: (a) YOLOv8n (Nano), (b) YOLOv8s (Small), (c) YOLOv8m (Medium), y (d) YOLOv8l (Large). Cada conjunto de gráficos muestra la evolución de las métricas clave, como la precisión y la pérdida, a lo largo de las épocas de entrenamiento para cada versión de YOLOv8, destacando sus fortalezas y limitaciones en función de su tamaño y capacidad computacional.	73
5.27. Matrices de Confusión Normalizadas de los Modelos YOLOv8n y YOLOv8s .	74
5.28. Rendimiento de las métricas de entrenamiento del modelo YOLOv8n con el conjunto de imágenes reales CORSICAN.	76
5.29. Matriz de confusión del set de test con las predicciones del modelo YOLOv8n en el conjunto de datos de imágenes reales de CORSICAN.	77
5.30. Ejemplos de detecciones predichas por el modelo YOLOv8n entrenado con sólo imágenes reales de CORSICAN. Se observa cómo patrones de fuego reales que son grabados de manera cercana pueden ser detectados pero imágenes de otros entornos no es capaz de distinguirlo.	78
5.31. Rendimiento de las métricas de entrenamiento del modelo YOLOv8n con el conjunto de imágenes mezcladas.	79
5.32. Matriz de confusión del set de test con las predicciones del modelo YOLOv8n en el conjunto de datos de imágenes mezcladas.	79
5.33. Ejemplos de detecciones predichas por el modelo YOLOv8n entrenado con imágenes reales de CORSICAN y generadas sintéticamente con Juggernaut XL. Se observa cómo patrones de fuego reales que son grabados de manera cercana pueden ser detectados y algunas imágenes de otros entornos puede llegar a detectar humo y fuego.	80

Índice de cuadros

1.1. Grado de relación del TFG con los Objetivos de Desarrollo Sostenible.	5
1.2. Coste de Personal.	6
1.3. Coste de Recursos Hardware.	6
1.4. Coste de Software y Servicios.	7
1.5. Coste Total del Proyecto.	7
2.1. Comparación del rendimiento en clasificación en ImageNet (corte simple, modelo simple). Los resultados para VGG-16 y ResNet-152 se incluyen como referencia. La versión de Inception V3 evaluada no incluye la torre auxiliar. Fuente: Adaptado de Chollet (2017) [5].	12
2.2. Comparación del tamaño del modelo y velocidad de entrenamiento entre Inception V3 y Xception. Fuente: Adaptado de Chollet (2017) [5].	12
2.3. Comparativa de métricas de las diferentes versiones de YOLO	16
5.1. Evaluación del rendimiento del modelo U-Net personalizado en el conjunto de datos de incendios en RGB.	50
5.2. Evaluación del rendimiento del modelo U-Net personalizado en el conjunto de datos de incendios en IR.	54
5.3. Comparación del rendimiento total de todas las clases en la fase de evaluación de YOLOv8 para diferentes tamaños de modelo. Fuente: Datos obtenidos de los resúmenes de métricas.	74
5.4. Comparación del rendimiento de YOLOv8 en diferentes conjuntos de datos. Fuente: Datos obtenidos de los resúmenes de métricas.	80
5.5. Comparación de resultados con diferentes configuraciones de postprocesamiento en YOLOv8n. Cada columna representa un ejemplo de detección de fuego y humo en cinco imágenes distintas (Imagen 1 - Imagen 5) bajo diversas configuraciones de <i>IoU</i> y umbral de confianza. Observamos que reducir el <i>IoU</i> o aumentar el umbral de confianza puede ayudar a consolidar y afinar las detecciones, optimizando la precisión en escenarios específicos.	82

Capítulo 1

Introducción

La detección temprana de incendios es un factor crítico en la protección de infraestructuras, tanto en áreas industriales como en zonas naturales. En entornos portuarios, donde confluyen diversas actividades de alta complejidad, el riesgo de incendios se ve incrementado por la presencia de grandes volúmenes de materiales inflamables, el movimiento constante de vehículos y mercancías, y las condiciones ambientales adversas que pueden dificultar la intervención rápida. Estos riesgos no solo ponen en peligro las instalaciones y los bienes materiales, sino también la vida de las personas y la estabilidad económica de las regiones cercanas.

A pesar de los avances en la tecnología de detección de incendios, los sistemas tradicionales suelen presentar limitaciones en contextos amplios y dinámicos, como los puertos y zonas marítimas. Factores como la niebla, el humo o el clima adverso pueden reducir la efectividad de los sistemas convencionales. En este contexto, las cámaras térmicas ofrecen una solución más robusta, ya que su capacidad para detectar el calor emitido por los objetos permite identificar focos de calor anómalos, como los causados por incendios, incluso en condiciones de baja visibilidad.

Este Trabajo Fin de Grado (TFG) se centra en el desarrollo de un modelo basado en técnicas de aprendizaje automático para la detección de anomalías térmicas, aplicable a una variedad de entornos donde el riesgo de incendios es alto. Aunque el enfoque principal de la investigación ha sido el entorno portuario y marítimo, el modelo está diseñado para ser adaptable a otros tipos de áreas extensas y complejas. Se ha utilizado una cámara térmico-óptica Hikvision bi-espectro, enfocando el análisis en la parte térmica, que es menos susceptible a factores ambientales adversos. La cámara genera termogramas que permiten la detección de fuentes de calor inusuales.

El proyecto ha sido desarrollado en colaboración con la empresa QUALITAS ARTIFICIAL INTELLIGENCE AND SCIENCE (QAISC) y la PLATAFORMA OCEÁNICA DE CANARIAS (PLOCAN), lo que ha facilitado el acceso a imágenes térmicas y equipos avanzados para realizar el desarrollo e investigación. Esta colaboración ha sido clave en la obtención de datos de alta calidad, necesarios para entrenar y optimizar el modelo propuesto.

Sin embargo, se presentaron desafíos importantes durante el desarrollo del proyecto, como la falta de acceso a datos RAW y la necesidad de simular incendios en las imágenes debido a la escasez de datos reales de fuego en escenarios portuarios o marítimos. Estas limitaciones han requerido soluciones creativas, utilizando datasets sintéticos para entrenar la red neuronal, lo que ha permitido obtener resultados precisos a pesar de las restricciones iniciales.

El objetivo principal de este trabajo ha sido desarrollar un sistema eficiente y robusto para la detección temprana de incendios, lo que contribuye tanto a la prevención de daños materiales como a la protección de vidas humanas. Durante el desarrollo, se investigaron y probaron diferentes enfoques basados en técnicas de aprendizaje profundo, con el fin de identificar el método más adecuado para la detección de anomalías térmicas. A lo largo del proceso, se emplearon tanto imágenes sintéticas como reales para entrenar y validar el modelo, lo que permitió optimizar su rendimiento y garantizar su precisión en escenarios reales de detección de incendios.

Por lo tanto, este TFG no solo demuestra la aplicación de conocimientos adquiridos durante el Grado, sino que también representa una contribución significativa al campo de la seguridad en la detección temprana de incendios en áreas críticas. La colaboración con QAISC y PLOCAN ha sido determinante para el desarrollo exitoso de esta solución innovadora, que tiene el potencial de ser implementada en una amplia gama de escenarios donde la prevención de incendios es crucial, más allá de los entornos portuarios.

1.1. Objetivos del proyecto

Este proyecto aborda la necesidad crítica de mejorar la detección de incendios en entornos portuarios y marítimos mediante el uso de tecnologías avanzadas. Se ha planteado desarrollar un modelo de aprendizaje automático que detecte de manera eficaz áreas de posibles incendios. Para alcanzar este propósito, se han definido una serie de objetivos específicos, detallados a continuación, que garantizan la creación de un sistema robusto y eficiente.

1. **Desarrollo de un Modelo de Aprendizaje Automático:** diseñar y desarrollar un modelo avanzado de aprendizaje automático que sea capaz de procesar y analizar termogramas obtenidos mediante una cámara bi-espectro. Este modelo deberá identificar de manera eficiente y precisa puntos o zonas de calor en las imágenes, empleando arquitecturas como U-Net para la segmentación y Xception para la clasificación.
2. **Procesamiento y Análisis de Termogramas:** implementar técnicas avanzadas de procesamiento de imágenes para convertir los datos capturados por las cámaras térmicas en información utilizable. Esto incluye la manipulación de imágenes en escala de grises y la aplicación de tratamientos específicos sobre máscaras de calor, permitiendo un análisis detallado de las variaciones de temperatura.
3. **Validación y Pruebas en Entornos Reales:** probar y validar el modelo desarrollado utilizando los datos provistos por QAISC, capturados en el entorno del banco de ensayos de la Plataforma Oceánica de Canarias (PLOCAN). Se debe asegurar que el modelo es

robusto y eficaz en condiciones reales y diversas, incluyendo variaciones ambientales y diferentes escenarios marítimos.

4. **Integración de Conocimientos Técnicos y Aplicados:** integrar los conocimientos teóricos adquiridos durante el grado con habilidades prácticas en procesamiento de imágenes, aprendizaje automático y análisis de datos. Este objetivo busca consolidar la formación académica del estudiante con experiencias prácticas y aplicadas en un contexto real.
5. **Estado del Arte y Síntesis de Imágenes:** realizar una revisión exhaustiva del estado del arte en la detección de incendios y anomalías térmicas, identificando las técnicas y metodologías más avanzadas. Además, se debe desarrollar un proceso de síntesis de imágenes para recrear escenarios de incendios, utilizando datasets sintéticos que permitan entrenar y mejorar el modelo en ausencia de datos reales de incendios en embarcaciones.

En conjunto, estos objetivos apuntan a la creación de un sistema innovador y práctico para la detección temprana de incendios en entornos portuarios y marítimos, contribuyendo significativamente a la seguridad y prevención de daños en estas áreas. La colaboración con QAISC y PLOCAN proporciona los recursos y datos necesarios para el desarrollo y validación del modelo, asegurando su efectividad en condiciones reales y diversas.

1.2. Competencias Específicas

En el desarrollo de este proyecto, se han aplicado diversas competencias específicas [9] adquiridas durante el Grado en Ingeniería Informática. A continuación, se detallan estas competencias y cómo se han implementado en el contexto del proyecto:

CI1: Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.

En este proyecto, esta competencia se ha aplicado al desarrollar un modelo de aprendizaje automático para la detección de incendios en entornos portuarios. Se han seguido estándares de fiabilidad y seguridad, además de principios éticos y normativas vigentes para garantizar la calidad del sistema.

CI2: Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.

Esta competencia se refleja en la planificación detallada y dirección del proyecto en fases, con entregas y revisiones periódicas. Este enfoque ha permitido una mejora continua del sistema y una constante evaluación de su impacto económico y social en la seguridad portuaria.

CI6: Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.

En el diseño del modelo de aprendizaje automático, se implementaron algoritmos avanzados tras un análisis exhaustivo para seleccionar las arquitecturas U-Net y Xception, adecuadas para segmentar y clasificar imágenes térmicas.

CI7: Conocimiento, diseño y utilización de forma eficiente de los tipos y estructuras de datos más adecuados a la resolución de un problema.

El proyecto ha empleado un diseño eficiente de estructuras de datos para procesar las imágenes térmicas capturadas, utilizando termogramas y máscaras de calor para gestionar y analizar las variaciones de temperatura con precisión.

CI8: Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.

Esta competencia se ha demostrado al seleccionar paradigmas y lenguajes de programación como Python y librerías especializadas, asegurando que la aplicación desarrollada sea robusta, segura y eficiente.

CI15: Conocimiento y aplicación de los principios fundamentales y técnicas básicas de los sistemas inteligentes y su aplicación práctica.

Para aplicar esta competencia, se implementaron redes neuronales convolucionales y técnicas de procesamiento de imágenes en el modelo de aprendizaje automático, logrando una mejora significativa en la detección de incendios.

CI16: Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería del software.

En el desarrollo del proyecto, se siguieron principios y metodologías de la ingeniería del software, adoptando un enfoque iterativo e incremental con entregas de prototipos por fases y revisiones periódicas, asegurando la calidad y funcionalidad del sistema final.

1.3. Alineamiento con los Objetivos de Desarrollo Sostenible

El presente Trabajo de Fin de Grado (TFG), enfocado en la detección de anomalías térmicas mediante modelos de aprendizaje profundo, busca contribuir a la seguridad en distintos entornos y la gestión sostenible de los recursos. A continuación, se presenta el grado de alineamiento del proyecto con los ODS:

Cuadro 1.1: Grado de relación del TFG con los Objetivos de Desarrollo Sostenible.

ODS	0 No procede	1 Bajo	2 Medio	3 Alto
1 Fin de la Pobreza	X			
2 Hambre cero	X			
3 Salud y Bienestar				X
4 Educación de calidad	X			
5 Igualdad de género	X			
6 Agua limpia y saneamiento				X
7 Energía asequible y no contaminante	X			
8 Trabajo decente y crecimiento económico			X	
9 Industria, Innovación e Infraestructuras				X
10 Reducción de las desigualdades	X			
11 Ciudades y comunidades sostenibles			X	
12 Producción y consumo sostenibles				X
13 Acción por el clima				X
14 Vida submarina		X		
15 Vida de ecosistemas terrestres		X		
16 Paz, justicia e instituciones sólidas	X			
17 Alianzas para lograr objetivos	X			

1.3.1. Justificación del alineamiento

- **ODS 3: Salud y Bienestar** (Relación alta). El proyecto contribuye a prevenir incidentes en instalaciones críticas mediante la detección temprana de anomalías térmicas, lo que protege la salud y seguridad de las personas.
- **ODS 6: Agua limpia y saneamiento** (Relación alta). El monitoreo térmico puede ser útil para la gestión eficiente de recursos hídricos al detectar pérdidas o fugas en infraestructuras relacionadas con el agua.
- **ODS 8: Trabajo decente y crecimiento económico** (Relación media). Optimizar la gestión de infraestructuras mediante tecnología avanzada impulsa la eficiencia económica y genera nuevas oportunidades tecnológicas.
- **ODS 9: Industria, Innovación e Infraestructuras** (Relación alta). Este TFG fomenta la innovación tecnológica para el desarrollo de infraestructuras resilientes y sistemas de detección avanzada.
- **ODS 11: Ciudades y comunidades sostenibles** (Relación media). Los sistemas de detección térmica pueden aplicarse para mejorar la sostenibilidad y seguridad en comunidades urbanas.
- **ODS 12: Producción y consumo sostenibles** (Relación alta). La detección temprana de anomalías optimiza el uso de recursos, reduciendo el desperdicio y promoviendo prácticas sostenibles.

- **ODS 13: Acción por el clima** (Relación alta). Identificar riesgos térmicos ayuda a mitigar problemas relacionados con el cambio climático y mejorar la resiliencia ante desastres ambientales.
- **ODS 14: Vida submarina** (Relación baja). Aunque menos directa, la detección térmica podría contribuir en investigaciones o monitoreos en entornos acuáticos.
- **ODS 15: Vida de ecosistemas terrestres** (Relación baja). En contextos naturales, el sistema puede detectar anomalías térmicas relacionadas con incendios forestales, protegiendo ecosistemas terrestres.

1.4. Presupuesto

En esta sección se presenta el desglose del presupuesto necesario para la realización de este Trabajo de Fin de Grado (TFG). El presupuesto se divide en los siguientes conceptos principales: coste de personal, recursos hardware, software y servicios. Finalmente, se calculará el coste total del proyecto.

1.4.1. Coste de Personal

El coste de personal incluye el tiempo empleado por el alumno en el desarrollo del proyecto y el tiempo dedicado por los tutores en asesorías y revisiones.

Cuadro 1.2: Coste de Personal.

Personal	Número de horas	Precio/hora (€)	Coste (€)
Alumno	300	60	18000
Tutores	11	100	1100
Total			19100

1.4.2. Coste de Recursos Hardware

El coste de hardware incluye los equipos utilizados durante el desarrollo del proyecto, siendo estos un portátil y una cámara térmica.

Cuadro 1.3: Coste de Recursos Hardware.

Recurso	Coste (€)
Portátil HP 15s-fq2087ns, i5	68,625*
Cámara Hikvision PTZ térmico-óptica	59.033,64*
Total	59.102,265

Nota: El coste de la cámara es aproximado y no incluye los gastos de instalación. El portátil tiene un precio total de 549€ pero se ha ponderado por mes (6 meses, $549 * (6/48)$).

1.4.3. Coste de Software y Servicios

El coste de software y servicios incluye las suscripciones y servicios necesarios para el desarrollo del proyecto.

Cuadro 1.4: Coste de Software y Servicios.

Recurso	Coste (€)
Suscripción 1 mes de servicio Getimg.ai	11,00
Total	11,00

1.4.4. Coste Total del Proyecto

El coste total del proyecto se calcula sumando todos los costes anteriores.

Cuadro 1.5: Coste Total del Proyecto.

Concepto	Coste (€)
Coste de Personal	19100,00
Coste de Hardware	59.102,265
Coste de Software y Servicios	11,00
Total	78213,265

Capítulo 2

Estado actual

Actualmente, el aprendizaje profundo está revolucionando múltiples sectores gracias a su capacidad de aportar soluciones avanzadas para resolver problemas que presentan cierta complejidad y que anteriormente no podían ser resueltos de manera eficaz. Desde su auge en las últimas décadas, los avances en el desarrollo de arquitecturas de redes neuronales profundas y la posibilidad de acceder a grandes volúmenes de datos han permitido a esta tecnología florecer en algunos campos como la visión por computador, el procesamiento de lenguaje natural o la detección de anomalías. Estos avances no solo han mejorado el rendimiento en tareas tradicionales, sino que han abierto la puerta a nuevas aplicaciones que tienen un impacto significativo en la prevención de riesgos, la seguridad y la automatización de procesos.

En el contexto de la prevención de riesgos, el aprendizaje profundo ha demostrado ser extremadamente eficaz en la detección temprana de situaciones anómalas. Por ejemplo, en la industria de la aviación y la ingeniería civil, se han implementado redes neuronales convolucionales (CNN) para monitorear infraestructuras críticas y detectar fallos estructurales antes de que se conviertan en problemas graves [21]. De manera similar, en el sector de la salud, los sistemas basados en aprendizaje profundo pueden detectar anomalías en imágenes médicas, facilitando la identificación temprana de enfermedades como el cáncer [10].

En el ámbito de la detección de anomalías, uno de los campos más abundantes, el aprendizaje profundo ha permitido diseñar sistemas que pueden monitorear grandes volúmenes de datos en tiempo real y detectar patrones inusuales que indican posibles riesgos. Aplicaciones en la seguridad industrial incluyen la vigilancia mediante cámaras térmicas y sistemas de alerta temprana basados en redes neuronales para prevenir incendios, como lo demuestran estudios recientes en el uso de cámaras infrarrojas y redes neuronales convolucionales para la detección de incendios forestales [46]. Estos sistemas permiten identificar puntos calientes o combustiones incipientes, ofreciendo alertas tempranas que minimizan los daños y mejoran la seguridad.

Otra área en la que el aprendizaje profundo está teniendo un fuerte impacto es en la seguridad y vigilancia. Los sistemas de cámaras inteligentes alimentados por redes neuronales convolucionales y técnicas de detección de objetos, como YOLO (*You Only Look Once*), son capaces de reconocer comportamientos sospechosos o identificar intrusiones en tiempo real

en áreas sensibles, como instalaciones portuarias, zonas de almacenamiento de mercancías peligrosas o fábricas de alto riesgo. Estos sistemas están siendo implementados cada vez más para mejorar la seguridad perimetral y proteger instalaciones críticas [39].

A medida que estas tecnologías avanzan, también se están explorando enfoques más sofisticados, como las redes generativas adversarias (GANs) y los modelos autorregresivos, que permiten la simulación y predicción de escenarios potencialmente peligrosos. Estos modelos están siendo utilizados en sectores como la energía para predecir fugas o explosiones en plantas petroquímicas, y en la ciberseguridad, para identificar patrones de ataque antes de que se produzcan.

2.1. Clasificación

La clasificación ha sido una de las tareas principales dentro del campo del aprendizaje automático, y es considerada uno de los problemas más básicos y estudiados. El objetivo de la clasificación es asignar una o varias etiquetas a un conjunto de datos de entrada, que puede incluir imágenes, textos, señales u otros tipos de datos. En el contexto de la visión por computador, la clasificación implica tomar una imagen y determinar a qué clase pertenece a través de una etiqueta.

2.1.1. El perceptrón y el inicio de las redes neuronales

El estudio de la clasificación en el ámbito del aprendizaje automático tiene sus orígenes en los primeros trabajos sobre redes neuronales artificiales, con el desarrollo del **perceptrón** en la década de 1950 por **Frank Rosenblatt** [44]. El perceptrón es un modelo matemático que consiste en una sola capa de neuronas con una función de activación lineal, capaz de resolver problemas de clasificación binaria. Su principal objetivo era simular el comportamiento de las neuronas del cerebro humano para reconocer patrones en datos de entrada (Figura 2.1).

El perceptrón se convirtió en un modelo influyente en la investigación de las redes neuronales y permitió avances en la clasificación de patrones. Sin embargo, fue rápidamente evidente que su capacidad estaba limitada, ya que no podía resolver problemas no lineales, como la función XOR. Este obstáculo, junto con críticas de investigadores como Minsky y Papert en los años 70, frenó durante un tiempo el progreso en el campo de las redes neuronales [34].

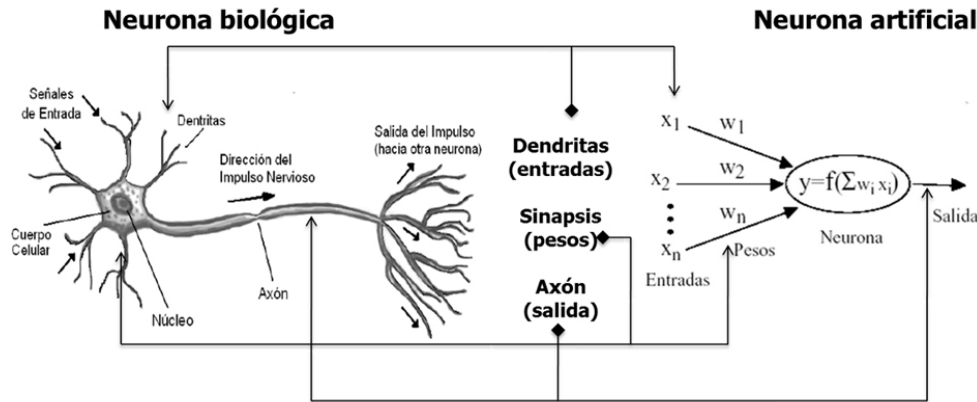


Ilustración 2.1: Comparación entre una neurona biológica y una neurona artificial. La imagen muestra las analogías estructurales entre ambos tipos de neuronas, destacando elementos como las dendritas (entradas), el cuerpo celular (procesamiento) y el axón (salida) en la neurona biológica, y sus equivalentes en la neurona artificial. Fuente: Adaptado de “Procedimiento para el pronóstico de la demanda mediante redes neuronales artificiales” por Lao León et al. (2017), Ciencias Holguín, 23(1), p. 5.

2.1.2. El renacimiento de las redes neuronales: el trabajo de LeCun y LeNet

Durante los años 80 y 90 se reanudó el avance de las redes neuronales con la introducción de nuevas técnicas de optimización, como el *backpropagation*, que permitió entrenar redes neuronales profundas de manera más eficiente. En este contexto, **Yann LeCun** desarrolló la **LeNet-5**, una arquitectura de red neuronal convolucional (CNN) que marcó un punto de inflexión en la clasificación de imágenes [27]. Esta arquitectura fue desarrollada principalmente para poder reconocer textos manuscritos mediante imágenes.

LeCun utilizó **convoluciones** para extraer características locales de las imágenes y **capas de pooling** (agrupamiento) para reducir la dimensionalidad, permitiendo que la red fuera más eficiente. Estas ideas sirvieron como base para las arquitecturas modernas de redes convolucionales, y mostraron por primera vez el poder de las CNN en tareas de clasificación de imágenes. El trabajo de LeCun resucitó el interés por las redes neuronales y consolidó las CNN como una herramienta esencial para problemas de visión por computador.

2.1.3. Tipos de clasificación

Dependiendo del tipo de problema a resolver, existen diferentes enfoques de clasificación:

- **Clasificación binaria:** Uno de los tipos más simples de clasificación, que categoriza los datos de entrada en dos clases, como “es fuego” o “no es fuego”. La salida suele representarse como **0** o **1**, donde 0 indica la ausencia de una condición y 1 su presencia. En este tipo de problemas, se utiliza comúnmente la función de activación sigmoide en

la capa de salida para obtener probabilidades entre 0 y 1, que luego se comparan con un umbral (habitualmente 0.5) para determinar la clase final.

- **Clasificación multiclase:** Se utiliza cuando los datos pueden pertenecer a una de varias categorías. En este caso, la red genera una probabilidad para cada clase y la clase con la mayor probabilidad es seleccionada. Este enfoque es particularmente útil en problemas donde se requiere una diferenciación compleja entre varias categorías. Para implementar la clasificación multiclase, se usa una capa **softmax** que convierte las salidas en una distribución de probabilidad sobre las clases.
- **Clasificación multietiqueta:** A diferencia de los otros enfoques, permite asignar múltiples etiquetas a una sola instancia. Esto es útil cuando una imagen contiene múltiples objetos o características, como “persona”, “árbol” y “perro”. En este caso, se usan funciones de activación sigmoide para cada etiqueta y se comparan con un umbral para decidir la asignación de múltiples etiquetas.

2.1.4. Modelos de clasificación basados en CNN

Las CNN han sido uno de los principales motores del progreso en la clasificación de imágenes. Como hemos visto con LeNet, estas redes utilizan **filtros** para detectar características locales en las imágenes y luego, a través de varias capas convolucionales, combinan estas características para reconocer patrones complejos. Modelos posteriores, como **AlexNet** [25], **VGG** [47] y **ResNet** [17], han mejorado las CNN tanto en profundidad como en precisión.

AlexNet, por ejemplo, fue el primer modelo que demostró el poder de las redes profundas para la clasificación de grandes conjuntos de datos, como **ImageNet** [7], utilizando hasta 8 capas de profundidad y técnicas como **dropout**, que resumidamente consiste en desconectar ciertas neuronas en cada entrenamiento para evitar el sobreajuste. Posteriormente, ResNet introdujo la idea de las conexiones residuales, permitiendo entrenar redes aún más profundas sin degradación del rendimiento.

2.1.5. Xception

Un modelo más reciente que ha mostrado excelentes resultados es **Xception** (Extreme Inception), desarrollado por **François Chollet** [5]. Xception está basado en la arquitectura **Inception**, originalmente presentada por Szegedy et al. [49], que introdujo un nuevo enfoque para mejorar las redes convolucionales profundas a través de *Inception modules*.

¿Qué es Inception?

Inception se diseñó con el objetivo de aprovechar el hecho de que diferentes tipos de convoluciones pueden capturar diferentes características de una imagen. En lugar de elegir una única operación (como una convolución de cierto tamaño de filtro) para aplicar en cada capa, Inception utiliza múltiples tamaños de filtros y combinaciones de operaciones en paralelo.

Cada *Inception block* realiza convoluciones de diferentes tamaños (por ejemplo, 1x1, 3x3, y 5x5), además de *pooling* (agrupamiento). La salida de cada una de estas operaciones se concatena, permitiendo que el modelo aprenda a capturar características a múltiples escalas sin aumentar de manera drástica la complejidad computacional.

Sin embargo, a pesar de las mejoras introducidas por Inception, aún existía la necesidad de optimizar el número de parámetros y aumentar la eficiencia en términos de cálculo.

¿Qué mejora Xception?

Xception lleva la idea de Inception un paso más allá al reemplazar los módulos de convolución tradicionales por **convoluciones separables en profundidad**. Estas convoluciones separan el proceso en dos fases: primero, una convolución espacial que aplica un filtro a cada canal de la entrada (como en las convoluciones tradicionales), y luego una convolución punto a punto (1x1) para combinar los canales. Esto descompone la operación convolucional de manera más eficiente, lo que permite reducir significativamente la cantidad de parámetros y los costos computacionales.

Al descomponer la convolución, se logra un modelo más simple y con menos parámetros, pero que sigue siendo capaz de aprender representaciones complejas y detalladas de los datos. Esto se traduce en un mejor rendimiento en tareas de clasificación de imágenes a gran escala, como en el conjunto de datos **ImageNet** [7]. De hecho, Xception ha demostrado superar a Inception en términos de precisión y eficiencia en diversos *benchmarks*.

Cuadro 2.1: Comparación del rendimiento en clasificación en ImageNet (corte simple, modelo simple). Los resultados para VGG-16 y ResNet-152 se incluyen como referencia. La versión de Inception V3 evaluada no incluye la torre auxiliar. Fuente: Adaptado de Chollet (2017) [5].

Modelo	Precisión Top-1	Precisión Top-5
VGG-16	71.5 %	90.1 %
ResNet-152	77.0 %	93.3 %
Inception V3	78.2 %	94.1 %
Xception	79.0 %	94.5 %

Cuadro 2.2: Comparación del tamaño del modelo y velocidad de entrenamiento entre Inception V3 y Xception. Fuente: Adaptado de Chollet (2017) [5].

Modelo	Número de parámetros	Pasos por segundo
Inception V3	23,626,728	31
Xception	22,855,952	28

Este modelo es un ejemplo de cómo las arquitecturas convolucionales han evolucionado hacia modelos más eficientes sin comprometer la capacidad de aprendizaje, ya que, si bien Inception fue un avance significativo al introducir operaciones paralelas de diferentes escalas, Xception ha demostrado un rendimiento superior a este en diversos conjuntos de datos, incluyendo ImageNet, destacándose en su capacidad para aprender representaciones más detalladas a menor costo computacional.

2.2. Detección de Objetos

La **detección de objetos** es una tarea fundamental en el campo del aprendizaje profundo y la visión por computador, que va más allá de la simple **clasificación de imágenes**. Mientras que en la clasificación el objetivo es asignar una etiqueta a la imagen completa, en la detección de objetos se busca identificar qué objetos están presentes y determinar su ubicación exacta en la imagen mediante **cajas delimitadoras** (*bounding boxes*) [40].

Esto implica resolver simultáneamente dos problemas: la **clasificación** y la **localización**. Un buen sistema de detección de objetos debe cumplir con varias características clave:

- **Localización precisa:** La detección no solo identifica qué objetos hay en la imagen, sino también dónde están. Esto se logra generando cajas delimitadoras que enmarcan cada objeto presente.
- **Detección simultánea:** Los modelos deben ser capaces de identificar múltiples objetos en una sola imagen, incluso si pertenecen a diferentes clases.
- **Escalabilidad:** Es crucial que los sistemas de detección puedan manejar objetos de diferentes tamaños y escalas, especialmente en aplicaciones como la detección en imágenes aéreas o vídeos de seguridad, donde los objetos pueden variar considerablemente en tamaño.
- **Velocidad y precisión:** En muchas aplicaciones de la vida real, como la conducción autónoma o la vigilancia, la rapidez es tan importante como la precisión. Modelos como **YOLO** son conocidos por su alto rendimiento en tiempo real, mientras que modelos más detallados como **Faster R-CNN** suelen ser más precisos a expensas de la velocidad [39, 40].

2.2.1. Evolución de los Modelos de Detección de Objetos

El campo de la detección de objetos ha experimentado una evolución notable en la última década, con una serie de avances importantes que han cambiado cómo se abordan los problemas de localización y clasificación simultáneas. Algunos hitos clave en esta evolución incluyen:

- **R-CNN (2014):** El modelo R-CNN (*Regions with CNN Features*), propuesto por Girshick et al. [14], fue uno de los primeros en aplicar redes neuronales convolucio-

nales (CNNs) para la detección de objetos. Aunque era preciso, tenía el inconveniente de ser extremadamente lento, ya que cada región de interés (RoI) se procesaba de manera independiente.

- **Fast R-CNN y Faster R-CNN (2015)**: Para abordar la lentitud de R-CNN, Girshick introdujo **Fast R-CNN**, que mejoró la velocidad al compartir convoluciones entre todas las regiones [13]. Posteriormente, **Faster R-CNN** fue propuesto por Ren et al. [40], introduciendo la red de propuestas de regiones (**RPN**), lo que permitió generar regiones candidatas de manera mucho más rápida y precisa.
- **YOLO (You Only Look Once, 2016)**: Este modelo fue un cambio de paradigma en la detección de objetos, ya que trataba la tarea como un problema de regresión en lugar de generar regiones candidatas. Propuesto por Redmon et al. [39], YOLO divide la imagen en una cuadrícula y predice las cajas y las clases de cada célula de la cuadrícula en una sola pasada por la red, lo que lo convierte en un modelo extremadamente rápido, ideal para aplicaciones en tiempo real.
- **SSD (Single Shot Multibox Detector, 2016)**: Similar a YOLO, SSD realiza detección en una sola pasada por la red, pero se diferencia en que predice cajas en múltiples escalas, utilizando diferentes capas de la red para detectar objetos de distintos tamaños [31]. Esto le otorga mayor robustez en escenarios donde los objetos varían significativamente en tamaño.

2.2.2. YOLO: You Only Look Once

El modelo **YOLO (You Only Look Once)** ha supuesto un avance revolucionario en el campo de la detección de objetos. A diferencia de métodos anteriores, como R-CNN y sus variantes, que dividen la tarea en varias etapas (propuestas de regiones y clasificación), esta arquitectura trata la detección como un único problema de regresión. En lugar de generar propuestas de regiones candidatas para ser clasificadas, la red neuronal divide la imagen en una cuadrícula y predice simultáneamente las cajas delimitadoras y las clases de los objetos presentes en cada celda en una sola pasada.

Desde su lanzamiento en 2016, el enfoque basado en YOLO ha sido objeto de múltiples mejoras. Entre las versiones más destacadas se encuentran **YOLOv3**, **YOLOv4**, **YOLOv5** y, más recientemente, **YOLOv8**, que introdujo avances en precisión, velocidad y eficiencia. Actualmente, **YOLOv8** es una de las versiones más rápidas y precisas, y sigue siendo ampliamente utilizada en aplicaciones de detección en tiempo real, aunque ya existen versiones posteriores, como YOLOv11. La implementación más reciente de YOLOv8 ha sido desarrollada y mantenida por *Ultralytics* [22].

Funcionamiento de YOLOv8

El funcionamiento de la versión 8 se mantiene fiel a los principios de las versiones anteriores, en cuanto a dividir la imagen en una cuadrícula $S \times S$ y predecir cajas delimitadoras

y clases para cada celda. Sin embargo, introduce mejoras notables en varias áreas clave:

- **Backbone:** YOLOv8 utiliza una arquitectura mejorada basada en *CSPDarknet*, que es una optimización del **Darknet** original introducido en versiones anteriores. Este **backbone** se destaca por su eficiencia en la extracción de características, empleando bloques de convolución más rápidos y profundos para mejorar tanto la velocidad como la precisión. La incorporación de bloques CSP (Cross-Stage Partial Network) ayuda a reducir la duplicación de gradientes y optimiza el flujo de información dentro de la red [51].
- **Anclajes adaptativos:** Una de las principales innovaciones de YOLOv8 es la eliminación de los anclajes predefinidos, permitiendo que el modelo ajuste sus predicciones de manera más dinámica. Este enfoque mejora considerablemente la precisión en la detección de objetos de distintas escalas, adaptándose automáticamente a las características del objeto y al tamaño de la imagen [12].
- **Predicción multinivel:** Al igual que en versiones anteriores, la red emplea características a múltiples niveles para mejorar la detección de objetos de diferentes tamaños. YOLOv8 se beneficia de una mejor integración de estas características mediante el uso de un **Path Aggregation Network (PAN)** [30], lo que mejora la capacidad del modelo para detectar objetos pequeños sin sacrificar la precisión en la detección de objetos grandes.

La arquitectura general de YOLOv8 se organiza en tres componentes principales: el **backbone**, que extrae las características de la imagen en múltiples niveles de detalle; el **neck**, que combina estas características utilizando un PAN para fusionar información; y la **head**, que realiza las predicciones finales de las cajas delimitadoras, las confianzas y las clases de los objetos detectados.

Métricas para evaluar la detección de objetos

El rendimiento en aproximaciones de detección de objetos se evalúa a través de varias métricas clave:

- **mAP@0.5 (Mean Average Precision at IoU=0.5):** Es una de las métricas más comunes para evaluar la precisión de los modelos de detección de objetos. Un valor de **IoU (Intersection over Union)** del 50 % indica que la caja predicha y la caja real tienen al menos una superposición del 50 %. Se calcula promediando la precisión a lo largo de diferentes umbrales de *recall* [11].
- **IoU (Intersection over Union):** Este valor mide la calidad de la predicción, calculando el grado de coincidencia entre las cajas predichas y las cajas reales de los objetos. Se define como la relación entre el área de intersección y el área de unión entre ambas cajas.

$$IoU = \frac{\text{Área de la intersección}}{\text{Área de la unión}}$$

- **FPS (Frames Per Second):** Se utiliza para medir el número de imágenes procesadas por segundo, un factor crucial en aplicaciones que requieren alta velocidad de procesamiento, como los sistemas de conducción autónoma o la videovigilancia [43].
- **Parámetros del modelo:** Representan el número de parámetros entrenables en millones. Las versiones más ligeras de YOLO, como YOLOv5 y YOLOv8, en sus configuraciones reducidas, permiten que el modelo sea implementado en dispositivos con recursos limitados, sin comprometer significativamente la precisión.

Comparativa de versiones de YOLO

A continuación, se presenta una comparativa de las métricas clave de las diferentes versiones de YOLO, destacando el progreso en precisión y velocidad con cada iteración:

Cuadro 2.3: Comparativa de métricas de las diferentes versiones de YOLO

Versión de YOLO	mAP@0.5	IoU	FPS	Parámetros (millones)
YOLOv3	55.3 %	0.71	45	61.5
YOLOv4	65.7 %	0.75	62	63.9
YOLOv5	68.9 %	0.76	70	7.5 (modelo pequeño)
YOLOv8	72.3 %	0.77	85	6.2 (modelo pequeño)

La más reciente versión, **YOLOv8**, sobresale por su capacidad de lograr un equilibrio óptimo entre precisión (**mAP@0.5**) y velocidad (**FPS**), además de reducir significativamente el número de parámetros, lo que lo convierte en una opción preferida para aplicaciones en tiempo real. Esto lo hace particularmente útil en sistemas autónomos y robótica, donde tanto la rapidez como la precisión son fundamentales [22].

2.3. Segmentación

La **segmentación de imágenes** es una tarea fundamental dentro del campo del aprendizaje profundo, cuyo objetivo es dividir una imagen en diferentes regiones o segmentos, donde cada píxel es asignado a una clase o categoría específica. A diferencia de la clasificación, que asigna una única etiqueta a toda la imagen, o la detección de objetos, que los localiza utilizando cajas delimitadoras, la segmentación proporciona una descripción mucho más detallada al etiquetar cada píxel. Esta precisión es especialmente importante en aplicaciones que requieren una interpretación detallada de la escena, como en el diagnóstico médico, la robótica y la conducción autónoma.

La segmentación puede clasificarse en dos tipos principales: la **segmentación semántica**, que agrupa píxeles que pertenecen a la misma clase sin diferenciar entre diferentes instancias de un objeto, y la **segmentación de instancias**, que además distingue entre múltiples objetos de la misma clase, proporcionando una etiqueta individual para cada uno.

2.3.1. Evolución de la Segmentación de Imágenes

Los primeros enfoques para la segmentación de imágenes utilizaban métodos basados en reglas, como el umbralado o la segmentación basada en grafos. Sin embargo, estos métodos eran limitados, especialmente en escenarios donde la variabilidad en las imágenes es alta. La llegada de las redes neuronales convolucionales (CNN) marcó un punto de inflexión en el campo. El uso de CNNs permitió la extracción jerárquica de características de las imágenes, mejorando significativamente la precisión y la generalización de los modelos de segmentación.

El trabajo pionero de Long et al. (2015) introdujo las **Fully Convolutional Networks (FCN)**, que adaptaron las CNNs para realizar segmentación a nivel de píxel, reemplazando las capas densas con convoluciones para mantener la estructura espacial de las imágenes [32]. Este avance sentó las bases para el desarrollo de modelos más sofisticados como **U-Net** y **Mask R-CNN**, que se describen más adelante.

2.3.2. Características de la Segmentación

La segmentación de imágenes presenta varias características que la distinguen de otras tareas de visión por computador, como la clasificación y la detección de objetos:

- **Etiqueta por píxel:** A diferencia de la clasificación, que proporciona una única etiqueta para toda la imagen, la segmentación asigna una etiqueta a cada píxel, lo que permite un análisis más detallado de la escena.
- **Precisión en los contornos:** Mientras que la detección de objetos utiliza cajas delimitadoras para localizar objetos, la segmentación proporciona una delineación precisa de los contornos de los objetos, lo que es fundamental en aplicaciones como el análisis de imágenes médicas o la visión en robótica.
- **Complejidad computacional:** La segmentación requiere una mayor capacidad de procesamiento en comparación con la clasificación o la detección, ya que implica la predicción de etiquetas a nivel de píxel para toda la imagen.

2.3.3. ¿Cómo Funciona la Segmentación?

La segmentación de imágenes se basa en redes neuronales convolucionales (CNN), adaptadas para procesar imágenes a nivel de píxel. El flujo general del proceso de segmentación involucra los siguientes pasos:

- **Extracción de características:** Al igual que en las tareas de clasificación y detección, el modelo comienza extrayendo características relevantes de la imagen a través de una serie de capas convolucionales.
- **Upsampling:** Tras la extracción de características, se utilizan técnicas de *upsampling* o deconvolución para restaurar la resolución original de la imagen. Esto es crucial ya que

la segmentación requiere una salida que coincida en tamaño con la imagen de entrada, permitiendo asignar etiquetas a cada píxel.

- **Asignación de etiquetas:** Finalmente, se asigna una etiqueta a cada píxel utilizando una función de activación, como *softmax*, que convierte las predicciones en probabilidades para las diferentes clases.

2.3.4. Métodos Comunes para la Segmentación

Existen varios métodos bien establecidos para la segmentación de imágenes, muchos de los cuales han surgido como avances significativos en la literatura de aprendizaje profundo. A continuación, se describen algunos de los más relevantes:

- **Fully Convolutional Networks (FCN):** Propuesto por Long et al. en 2015, este enfoque fue uno de los primeros en adaptar las CNNs para la segmentación. Las FCN eliminan las capas densas utilizadas en las CNNs tradicionales y las reemplazan por convoluciones, lo que permite al modelo manejar imágenes de cualquier tamaño y asignar etiquetas a nivel de píxel [32].
- **U-Net:** Introducido por Ronneberger et al. en 2015, U-Net es un modelo ampliamente utilizado en el campo de la segmentación médica. Su arquitectura en forma de U combina capas convolucionales para la extracción de características con capas de *upsampling*, que restauran la resolución original. Esto permite que el modelo integre tanto detalles finos como información abstracta para realizar predicciones precisas a nivel de píxel [42].
- **Mask R-CNN:** Basado en el modelo Faster R-CNN, Mask R-CNN extiende este enfoque para realizar segmentación de instancias. A diferencia de otros métodos, Mask R-CNN no solo detecta y clasifica objetos, sino que también proporciona una máscara de segmentación para cada instancia, lo que permite diferenciar entre múltiples objetos de la misma clase en una imagen [16].
- **DeepLab:** El modelo DeepLab, propuesto por Chen et al., emplea convoluciones dilatadas para ampliar el campo de visión de las capas convolucionales sin perder resolución espacial. DeepLab ha demostrado ser especialmente eficaz en escenarios donde los objetos tienen bordes complejos o donde existen múltiples escalas dentro de la misma imagen [4].

2.3.5. U-Net

El modelo **U-Net**, propuesto por Ronneberger et al. en 2015 [42], es una red neuronal convolucional diseñada específicamente para la segmentación de imágenes biomédicas. U-Net ha sido ampliamente utilizado en diversas aplicaciones de segmentación debido a su capacidad para generar segmentaciones precisas a nivel de píxel, incluso cuando el tamaño del conjunto de datos es relativamente pequeño. Este modelo ha sido crucial en áreas como la segmentación

de células y tejidos en imágenes médicas, y su uso se ha expandido a otros campos de visión por computador.

Arquitectura

El nombre “U-Net” proviene de su característica arquitectura en forma de “U”. Esta arquitectura se compone de dos partes principales:

- **Camino de contracción (encoder):** Esta primera parte es una red convolucional típica que extrae características relevantes de la imagen a través de varias capas convolucionales, seguidas de capas de *max-pooling* que reducen la resolución espacial. Esta parte de la red actúa como un extractor de características que permite al modelo aprender representaciones más abstractas a medida que se desciende por la red.
- **Camino de expansión (decoder):** Después de la contracción, la segunda parte del modelo consiste en un proceso de *upsampling* que restaura la resolución original de la imagen, combinando características de las capas correspondientes del camino de contracción. Este proceso de combinación mediante conexiones de salto (*skip connections*) permite que la red utilice tanto la información de bajo nivel (detalles finos) como la información de alto nivel (características abstractas) para generar segmentaciones detalladas y precisas.

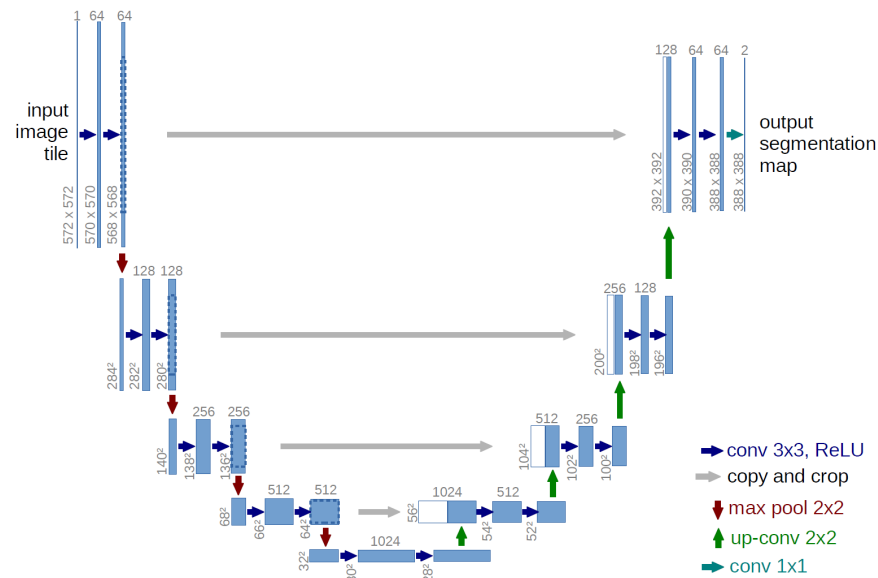


Ilustración 2.2: Arquitectura de U-Net. Cada bloque azul representa un mapa de características multi-canal, con el número de canales indicado en la parte superior de cada bloque. Las dimensiones espaciales x - y se encuentran en la esquina inferior izquierda. Las cajas blancas representan mapas de características copiados mediante las conexiones de salto (*skip connections*). Las flechas denotan las diferentes operaciones, como convoluciones, *max pooling*, y convoluciones transpuestas. Fuente: Adaptado de Ronneberger et al. (2015) [42].

En términos más simples, U-Net realiza primero una reducción de la resolución para aprender características generales, y luego restaura la resolución original para realizar predicciones a nivel de píxel. Las conexiones de salto son clave para evitar la pérdida de detalles durante el proceso de reducción de la imagen.

Funcionamiento

El flujo de trabajo de U-Net puede describirse en los siguientes pasos:

1. **Entrada de imagen:** La red recibe una imagen de entrada, típicamente de tamaño 572×572 píxeles en la implementación original.
2. **Extracción de características:** A través del camino de contracción, se extraen características de la imagen utilizando varias capas de convolución y *max-pooling*, lo que reduce el tamaño espacial de la imagen y aumenta la profundidad de las características.
3. **Reconstrucción de la imagen:** En el camino de expansión, se utilizan capas de deconvolución o *upsampling* para aumentar la resolución de las características, combinando esta información con las características del camino de contracción mediante las conexiones de salto.
4. **Salida de la segmentación:** El modelo genera un mapa de segmentación de salida, donde cada píxel se clasifica en una categoría, proporcionando una segmentación precisa.

Motivación y Aplicación

U-Net fue diseñado inicialmente para el desafío de segmentación en imágenes biomédicas, específicamente en la segmentación de células en microscopía de fluorescencia [42]. En ese contexto, las imágenes tienden a tener una alta variabilidad en cuanto a tamaño y forma de los objetos, lo que requiere un modelo capaz de adaptarse a esta variabilidad sin necesitar grandes cantidades de datos etiquetados.

Una de las principales ventajas de U-Net es su eficiencia en el uso de datos. Debido a su arquitectura con conexiones de salto, este modelo puede entrenarse con un número relativamente pequeño de imágenes, lo cual es fundamental en áreas como la medicina, donde la anotación de imágenes es costosa y laboriosa. Además, la red puede procesar imágenes completas de una vez, sin necesidad de dividir las imágenes en parches, lo que la hace especialmente efectiva para imágenes con estructuras finas.

Aparte de su uso en la segmentación médica, U-Net ha sido adaptado con éxito a otras aplicaciones, como la segmentación de imágenes satelitales, el procesamiento de imágenes de drones, y la detección de defectos en imágenes industriales.

Resultados y Estadísticas

Desde su introducción, U-Net ha mostrado resultados sobresalientes en múltiples desafíos de segmentación. En el desafío de la ISBI (International Symposium on Biomedical Imaging) de 2015, U-Net logró superar significativamente a otros modelos tradicionales, con una precisión de segmentación cercana al 92 % en la segmentación de membranas celulares [42].

Debido a su diseño, U-Net ha demostrado ser especialmente efectivo en la segmentación de imágenes donde los objetos tienen bordes complejos o estructuras pequeñas, como es el caso de las imágenes biomédicas. Su capacidad para equilibrar la extracción de características a gran escala y la preservación de detalles finos lo ha convertido en uno de los modelos más utilizados en el campo de la segmentación de imágenes.

2.4. Generative Adversarial Networks (GANs) y sus aplicaciones en la Transformación de Imágenes

Las **Generative Adversarial Networks (GANs)** son una clase de modelos de aprendizaje profundo propuestos por Ian Goodfellow et al. en 2014 [15]. Las GANs consisten en dos redes neuronales que compiten entre sí en un proceso de aprendizaje conjunto: una *red generadora (generator)* que intenta producir datos falsos que parezcan reales, y una *red discriminadora (discriminator)* que intenta distinguir entre los datos reales y los generados. El objetivo del generador es engañar al discriminador para que no pueda distinguir entre datos reales y sintéticos, mientras que el discriminador intenta mejorar su capacidad de detección.

Este enfoque ha sido muy exitoso en diversas aplicaciones de generación de datos sintéticos, siendo especialmente útil en la generación y transformación de imágenes, donde las GANs han mostrado resultados impresionantes.

2.4.1. Estructura y Funcionamiento de las GANs

La estructura básica de una GAN se basa en la interacción entre las dos redes mencionadas:

1. **Generador:** Recibe una entrada, generalmente un vector de ruido, y genera una imagen o dato sintético. Su objetivo es aprender la distribución de los datos reales y producir muestras que se asemejen a las del conjunto de datos real.
2. **Discriminador:** Toma como entrada tanto imágenes reales como generadas y clasifica si cada imagen es real o falsa. Se entrena para mejorar su capacidad de distinguir entre los datos reales y los falsos.

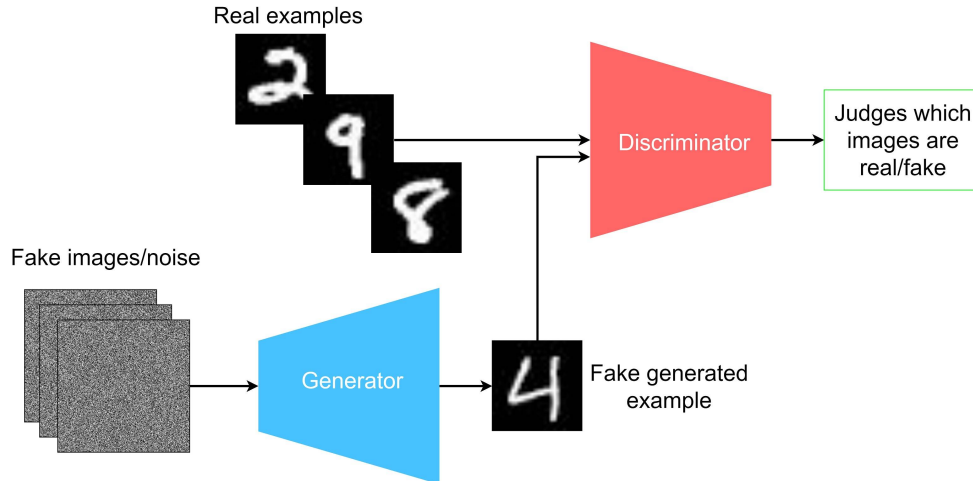


Ilustración 2.3: Esquema de funcionamiento de una Red Generativa Antagónica (GAN). En este diagrama, la red generadora (en azul) produce ejemplos falsos a partir de ruido o imágenes iniciales aleatorias. Estos ejemplos generados se envían al discriminador (en rojo), el cual también recibe ejemplos reales del conjunto de datos de entrenamiento. El discriminador evalúa si cada imagen es real o generada, proporcionando retroalimentación a la red generadora para mejorar sus resultados con el tiempo. Fuente: Adaptado de IBM Developer <https://developer.ibm.com/articles/generative-adversarial-networks-explained/>.

El entrenamiento de GANs puede verse como un juego minimax entre ambas redes, donde el generador intenta minimizar la habilidad del discriminador para distinguir entre imágenes reales y generadas, y el discriminador intenta maximizar esa misma habilidad. Matemáticamente, este proceso se modela mediante una función de costo basada en la teoría de juegos:

$$\min_G \max_D V(D, G) = E_{x \sim p_{\text{data}}(x)} [\log D(x)] + E_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

Donde $G(z)$ es la salida del generador y $D(x)$ es la salida del discriminador.

2.4.2. Aplicaciones y Evolución de las GANs

Desde su creación, las GANs han sido aplicadas en un gran número de tareas, incluyendo la generación de imágenes, la mejora de la resolución de imágenes (*super-resolution*), la síntesis de imágenes a partir de bocetos o mapas de etiquetas (*image-to-image translation*), y la creación de imágenes hiperrealistas.

A lo largo de los años, se han propuesto muchas variantes de las GANs para mejorar su estabilidad de entrenamiento y ampliar su aplicación a tareas más específicas. Algunas de estas variantes se centran en la traducción de imágenes entre dominios, lo que ha dado lugar a una gran variedad de modelos.

2.4.3. CycleGAN

El modelo **CycleGAN**, introducido por Zhu et al. en 2017 [55], es una variante de las GANs diseñada para realizar la traducción no supervisada entre dominios de imágenes. Mientras que muchos modelos de traducción de imágenes requieren pares de imágenes entrenadas de dos dominios (por ejemplo, una imagen real y su correspondiente imagen editada), CycleGAN es capaz de aprender esta transformación sin necesitar dichas parejas.

El principal avance de CycleGAN es el uso de la **consistencia cíclica** como restricción. Esto significa que si una imagen se transforma de un dominio a otro y luego se revierte, la imagen resultante debe ser idéntica a la imagen original. El modelo utiliza dos generadores y dos discriminadores: uno para la transformación de ida y otro para la transformación inversa, asegurando la consistencia.

Matemáticamente, la pérdida de consistencia cíclica se expresa como:

$$\mathcal{L}_{cyc}(G, F) = E_{x \sim p_{data}(x)}[\|F(G(x)) - x\|_1] + E_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1]$$

Esto permite transformar imágenes entre dominios completamente diferentes, como convertir fotografías de paisajes en pinturas al estilo de Van Gogh, o convertir imágenes de caballos en cebras.



Ilustración 2.4: Ejemplos del funcionamiento de CycleGAN en la traducción de imágenes sin pares de correspondencia. Cada fila muestra distintas aplicaciones de traducción de imágenes: de izquierda a derecha, se muestran la imagen de entrada x , la imagen generada $G(x)$ tras la traducción y la imagen reconstruida $F(G(x))$. Los ejemplos ilustran diversas tareas de traducción: de fotos a estilo Cezanne, de caballos a cebras, de invierno a verano en Yosemite, y de fotos aéreas a mapas de Google. La red generativa antagónica con consistencia cíclica permite transformar imágenes de un dominio a otro y volver al original manteniendo coherencia. Fuente: Adaptado de Zhu et al. (2017) [55].

2.4.4. Pix2Pix

El modelo **Pix2Pix**, desarrollado por Isola et al. en 2017 [20], es una arquitectura de GAN diseñada específicamente para la traducción de imágenes entre dominios emparejados con supervisión. A diferencia de CycleGAN, que permite la traducción sin correspondencias exactas entre dominios, Pix2Pix requiere un conjunto de datos alineado. Esto significa que cada imagen de un dominio debe tener una contraparte en el dominio objetivo, como en el caso de una foto y su mapa de etiquetas correspondiente o un boceto y su versión coloreada.

Pix2Pix utiliza una función de pérdida compuesta, que combina la pérdida adversaria típica de las GANs con una pérdida de consistencia basada en la $L1$ loss, la cual mide la diferencia entre la imagen generada y la imagen real emparejada. La función de pérdida total para el generador G se expresa como:

$$\mathcal{L}_{L1}(G) = E_{x,y}[\|y - G(x)\|_1]$$

Esta combinación permite a Pix2Pix no solo aprender a generar imágenes que "parezcan realistas, sino también a preservar características específicas de la imagen de entrada, asegurando una mayor fidelidad en la traducción.

Pix2Pix se ha aplicado exitosamente en diversas tareas de traducción de imágenes, tales como la conversión de bordes a fotografías realistas, mapas de etiquetas a imágenes fotorrealistas, y la adaptación de imágenes en diferentes condiciones de iluminación o estaciones del año. Su enfoque de supervisión permite un control más preciso sobre el estilo y el contenido de la imagen generada en comparación con modelos no supervisados.

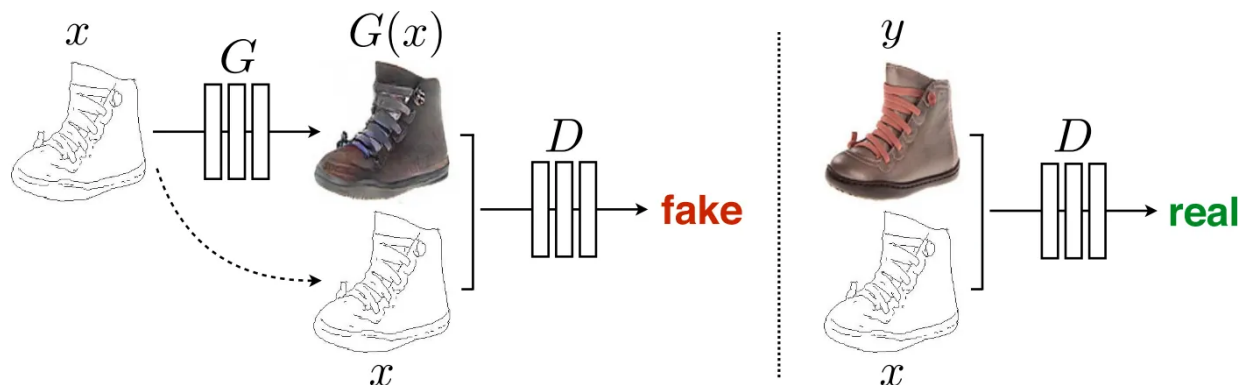


Ilustración 2.5: Proceso de entrenamiento de una GAN condicional en Pix2Pix para mapear contornos a fotografías. El generador G toma una imagen de contorno x y produce una imagen fotográfica $G(x)$. El discriminador D recibe pares de imágenes {contorno, foto} y aprende a diferenciar entre pares reales (del conjunto de entrenamiento) y pares sintéticos (creados por G). A diferencia de una GAN estándar, tanto el generador como el discriminador en Pix2Pix reciben la imagen de entrada como condicionante. Fuente: Adaptado de Isola et al. (2017) [20].

2.4.5. MUNIT

MUNIT (Multimodal UNsupervised Image-to-Image Translation), es un modelo de traducción de imágenes, desarrollado por Huang et al. en 2018 [18], que permite una generación multimodal no supervisada. A diferencia de otros enfoques como CycleGAN, que produce una única salida por cada entrada, MUNIT es capaz de generar múltiples posibles versiones de una imagen en el dominio objetivo al modelar de manera explícita el espacio latente, separando el contenido y el estilo de las imágenes.

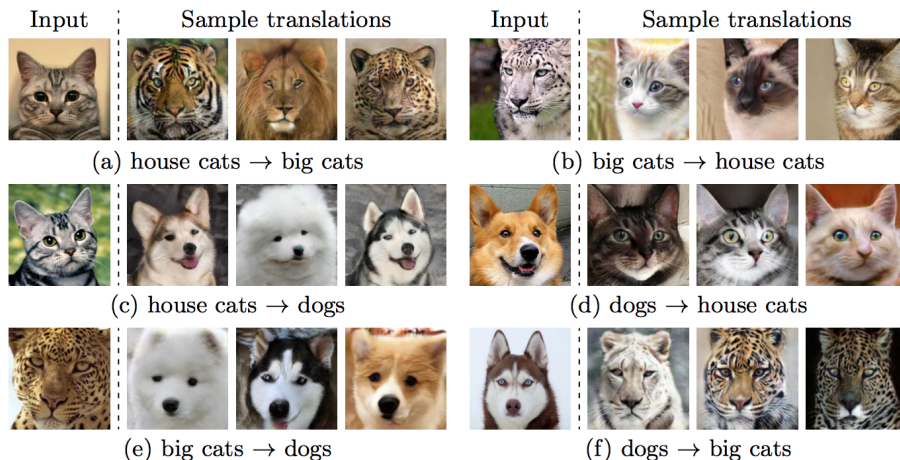


Ilustración 2.6: Ejemplos de traducción de imágenes utilizando el modelo MUNIT. La imagen muestra transformaciones estilísticas entre diferentes clases de animales, con la imagen de entrada en la columna “Input” y las traducciones en la columna “Sample translations”. MUNIT permite realizar estas traducciones manteniendo las características de contenido pero modificando el estilo. Fuente: Adaptado de Huang et al. (2018) [18].

La arquitectura de MUNIT se compone de dos codificadores y dos decodificadores, uno para cada dominio, que permiten la separación explícita del contenido y el estilo de las imágenes. El codificador de contenido abstrae las características invariantes del objeto, mientras que el codificador de estilo extrae las propiedades visuales específicas del dominio, como colores o texturas. Al combinar un código de contenido con diferentes códigos de estilo, MUNIT puede generar múltiples estilos para una misma imagen de entrada en el dominio objetivo.

La función de pérdida en MUNIT incluye términos para la consistencia cíclica (asegurando que la imagen pueda reconstruirse en su dominio original) y una pérdida de consistencia latente que preserva la coherencia en las representaciones de contenido a través de las transformaciones. Esta arquitectura permite al modelo retener características clave de la imagen de entrada mientras varía su apariencia, permitiendo así una traducción estilística flexible entre dominios.

MUNIT ha demostrado ser particularmente útil en aplicaciones donde es necesario traducir imágenes entre dominios con gran variabilidad estilística, sin la necesidad de datos emparejados.

2.5. Modelos de Procesamiento de Texto a Imagen

La generación de imágenes a partir de descripciones textuales ha experimentado un avance significativo, gracias al desarrollo de modelos generativos basados en aprendizaje profundo. Este tipo de procesamiento permite crear imágenes personalizadas y controladas a partir de texto, lo cual resulta útil en diversas aplicaciones como el arte, el diseño gráfico, la simulación y la accesibilidad. Los avances en arquitecturas de redes generativas antagónicas (GANs), modelos de difusión y transformadores (Transformers) han sido clave para esta evolución.

2.5.1. Origen y Evolución

El desarrollo de modelos de texto a imagen comenzó con la intención de unir las representaciones textuales y visuales. Los primeros modelos, como **StackGAN** [53], se basaban en Redes Generativas Antagónicas (GANs) condicionadas por texto y generaban imágenes en varias etapas para mejorar la coherencia visual. Posteriormente, modelos como **DALL-E** de OpenAI [38] utilizaron la arquitectura Transformer, elevando la capacidad para crear imágenes detalladas y coherentes a partir de texto. Los modelos de **Difusión**, como *Imagen* [45], perfeccionaron aún más este proceso mediante la eliminación progresiva de ruido condicionado por el texto, logrando imágenes más realistas y detalladas.

2.5.2. Arquitectura de los Modelos de Texto a Imagen

Los modelos de generación de imágenes a partir de texto modernos suelen contar con una estructura de tres componentes clave:

- ✓ **Encoder de Texto:** Este módulo convierte el texto en una representación latente que captura el significado semántico. Normalmente se emplean transformadores como GPT o BERT, que permiten al modelo comprender la relación entre las palabras y los elementos visuales en las imágenes.
- ✓ **Decoder de Imagen:** A partir de la representación latente del texto, el decoder genera la imagen condicionada a este. En los modelos de difusión, el decoder comienza con ruido aleatorio y refina la imagen progresivamente a través de múltiples pasos de inferencia.
- ✓ **Condicionamiento Adicional:** Algunos modelos incluyen mecanismos adicionales, como **IP-Adapter** [3] y **ControlNet** [54], que permiten un mayor control sobre la generación de la imagen, agregando señales estructurales adicionales, como bocetos o poses.

2.5.3. Modelos Sobresalientes

Algunos de los modelos más destacados en la generación de imágenes a partir de texto incluyen:

Juggernaut XL

El modelo **Juggernaut XL**, desarrollado por RunDiffusion y alojado en la plataforma Hugging Face ¹, utiliza una arquitectura de difusión avanzada que permite crear imágenes detalladas y coherentes a partir de texto. Juggernaut XL destaca por su capacidad para traducir descripciones complejas en imágenes realistas, siendo especialmente útil en aplicaciones comerciales y creativas. Este modelo emplea una arquitectura profunda que ajusta los detalles de la imagen en múltiples etapas, comenzando con ruido aleatorio y refinando gradualmente los elementos visuales de acuerdo con el texto.

DALL-E

DALL-E, desarrollado por OpenAI [38], fue uno de los primeros modelos en aplicar arquitecturas de Transformer para la generación de imágenes a partir de texto. DALL-E permite la creación de imágenes detalladas y coherentes, incluyendo conceptos abstractos y combinaciones inusuales de objetos. Su impacto ha sido significativo, ya que introdujo el uso de atención en la generación de imágenes, mejorando la coherencia semántica en las imágenes generadas y demostrando el potencial de los transformadores en esta tarea.

Stable Diffusion

Stable Diffusion [41], desarrollado por Stability AI, es ampliamente utilizado debido a su naturaleza de código abierto y su eficiencia. Utiliza una arquitectura de difusión para generar imágenes con menor número de pasos de inferencia, lo que permite lograr resultados de alta calidad sin necesidad de recursos computacionales excesivos. Su flexibilidad y accesibilidad han facilitado su adopción en múltiples comunidades y proyectos personalizados en una variedad de contextos.

2.5.4. Herramientas y Mecanismos Complementarios

Existen varias herramientas y mecanismos complementarios que permiten un control adicional sobre la generación de imágenes a partir de texto, mejorando la precisión y adaptabilidad de los modelos.

IP-Adapter

El **IP-Adapter** [3] permite introducir información adicional en el proceso de generación, como colores, estilo o poses. Esta herramienta facilita la modificación de ciertos elementos visuales, añadiendo una capa de control sobre el resultado final que es útil en aplicaciones que requieren características específicas en la imagen generada.

¹Disponible en: <https://huggingface.co/RunDiffusion/Juggernaut-X-v10>

ControlNet

ControlNet [54] permite incorporar señales estructurales, como contornos o poses predefinidas, para guiar el diseño y disposición de los elementos en la imagen final. Esta herramienta resulta útil cuando se necesita precisión en la disposición de objetos y en la configuración visual de la imagen, como en ilustraciones o planos arquitectónicos.

Profundidad y Mecanismos de Control de Profundidad

Incorporar información de **profundidad** mejora el sentido tridimensional en la imagen generada, lo cual es esencial en aplicaciones de simulación y realidad aumentada [2]. El uso de mecanismos de control de profundidad permite que los modelos representen detalles complejos con mayor precisión, logrando una percepción mejorada de distancia y volumen en los objetos.

2.5.5. Samplers en la Generación de Imágenes

Los *samplers* controlan la calidad y velocidad del proceso de generación de imágenes en los modelos de difusión. Algunos de los samplers más comunes incluyen:

- **DDIM (Denoising Diffusion Implicit Models):** Este sampler reduce el número de pasos de generación sin perder calidad en la imagen final.
- **Euler y Euler Acelerado:** Ajustan los pasos de difusión para lograr un balance entre precisión y velocidad, útiles para generar imágenes detalladas en menos tiempo.
- **PLMS (Predictive Linear Multi-Step):** Optimiza la generación mediante pasos predictivos, ofreciendo un control preciso sobre la calidad de la imagen final.

2.5.6. Aplicaciones y Usos Actuales

Los modelos de procesamiento de texto a imagen tienen aplicaciones en diversas áreas:

- **Arte Generativo:** Permiten la creación de obras de arte personalizadas a partir de descripciones textuales, democratizando el acceso a la creatividad visual.
- **Diseño Gráfico:** Utilizados para generar imágenes en publicidad, moda y medios, sin requerir un diseñador gráfico.
- **Visualización de Conceptos Complejos:** Útiles en arquitectura y cine, permitiendo a los usuarios visualizar ideas abstractas o complejas.
- **Accesibilidad:** Facilitan la creación de representaciones visuales accesibles para personas con discapacidades visuales, mejorando la comprensión de conceptos visuales.

Capítulo 3

Metodología de Trabajo

Se adoptó una metodología ágil basada en objetivos, complementada con el uso de herramientas de gestión visual de proyectos. Esta combinación permitió un enfoque flexible, adaptable y efectivo, lo cual fue fundamental para enfrentar los desafíos propios de un proyecto de investigación y desarrollo en el ámbito de la inteligencia artificial y el aprendizaje profundo.

3.1. Metodología Basada en Objetivos

La metodología basada en objetivos utilizada en este proyecto sigue los principios de la *Gestión por Objetivos* (*Management by Objectives, MBO*), un enfoque introducido por Peter Drucker en la década de 1950. MBO se basa en la definición clara de objetivos específicos y alcanzables, que se alinean con los resultados esperados en cada fase del proyecto. Este enfoque ha demostrado ser particularmente útil en proyectos de investigación, donde las etapas requieren evaluaciones continuas y ajustes dinámicos para alcanzar los objetivos establecidos [8].

Los principios clave de esta metodología incluyen:

1. **Definición clara de objetivos:** Para cada fase del proyecto, se establecieron metas específicas que orientaron el trabajo. Estos objetivos sirvieron como guía para el desarrollo, asegurando que cada paso tuviera un propósito definido y medible [19].
2. **Revisión periódica del progreso:** A lo largo del proyecto, se llevaron a cabo reuniones con los tutores, en las que se revisaban los avances y se discutían los resultados obtenidos. Estas sesiones de revisión permitieron validar el trabajo realizado y ajustar el enfoque para las siguientes fases [8].
3. **Flexibilidad y adaptabilidad:** Dado el carácter exploratorio del proyecto, fue necesario ajustar los objetivos en función de los resultados y desafíos encontrados en cada etapa. Esta flexibilidad permitió que el proyecto evolucionara de manera coherente, incorporando nuevos aprendizajes y hallazgos [24].

4. **Enfoque iterativo:** La metodología no solo permitió una secuencia clara de etapas, sino que también fomentó la iteración en las fases clave. Esto significa que en ciertos momentos, se volvía a revisar y ajustar etapas anteriores en función de los avances y resultados obtenidos [8].

Las etapas del proyecto se organizaron de manera secuencial, pero con espacio para retroalimentación y mejoras en cada fase. Estas incluyeron:

1. *Revisión del estado del arte en la detección de incendios mediante cámaras térmicas.*
2. *Investigación y análisis de conjuntos de datos ya creados*, para evaluar su aplicabilidad.
3. *Desarrollo de métodos para generar un dataset sintético* adecuado para entrenar el modelo.
4. *Validación del conjunto de datos*, asegurando su precisión y relevancia para la tarea.
5. *Implementación y prueba de distintos métodos de aprendizaje profundo* aplicados a la detección térmica.
6. *Evaluación de los resultados*, comparando los métodos probados y ajustando el enfoque.
7. *Redacción de la memoria*, documentando los hallazgos y el proceso de desarrollo.

Este enfoque basado en objetivos, adaptado a la naturaleza iterativa de la investigación, permitió una organización clara del trabajo y la posibilidad de ajustar el enfoque según las necesidades emergentes del proyecto [8].

3.2. Implementación de Kanban con Trello

Para complementar la metodología basada en objetivos y gestionar de manera visual las tareas diarias, se implementó un sistema *Kanban* utilizando la plataforma *Trello*. Kanban es un método de gestión del trabajo que se originó en la industria manufacturera japonesa, particularmente en las fábricas de *Toyota* en la década de 1940 [23]. En japonés, “Kanban” significa “tarjeta visual”, y el sistema fue diseñado para mejorar la eficiencia mediante la visualización del flujo de trabajo y la reducción de desperdicios. Con el tiempo, este método se adaptó a otros sectores, incluidos los proyectos de desarrollo de software y la gestión de proyectos [36].

Los principios clave de Kanban aplicados en este proyecto fueron:

1. **Visualización del flujo de trabajo:** En Trello, las tareas fueron organizadas en un tablero digital que representaba visualmente el estado de cada actividad. Las columnas del tablero se dividieron en fases como “*Por hacer*”, “*En proceso*”, “*Revisión*” y “*Completado*”. Esta representación gráfica facilitó la comprensión del progreso del proyecto y permitió identificar rápidamente qué tareas estaban pendientes y cuáles estaban avanzando [50].

2. **Limitación del trabajo en curso** (*Work in Progress, WIP*): Uno de los principios fundamentales de Kanban es limitar la cantidad de tareas en cada columna o etapa para evitar la sobrecarga de trabajo. En este proyecto, se definieron límites para el número de tareas en curso, lo que permitió un enfoque más controlado y eficiente, asegurando que no se asumieran demasiadas tareas al mismo tiempo [1].
3. **Gestión del flujo**: Kanban se centra en la optimización del flujo de trabajo, asegurando que las tareas avancen de manera continua y sin interrupciones. Al visualizar el progreso en Trello, fue posible identificar posibles cuellos de botella o tareas que requerían atención urgente, lo que facilitó la toma de decisiones para resolver problemas rápidamente y mantener el flujo de trabajo estable [23].
4. **Mejora continua**: Una de las características más importantes de Kanban es su enfoque en la mejora continua. Durante las reuniones con los tutores, se revisaba la información proporcionada por el tablero de Trello para identificar áreas de mejora, ajustar el enfoque si era necesario, y refinar el proceso a medida que avanzaba el proyecto. Esta retroalimentación constante permitió optimizar el trabajo a lo largo del tiempo [52].

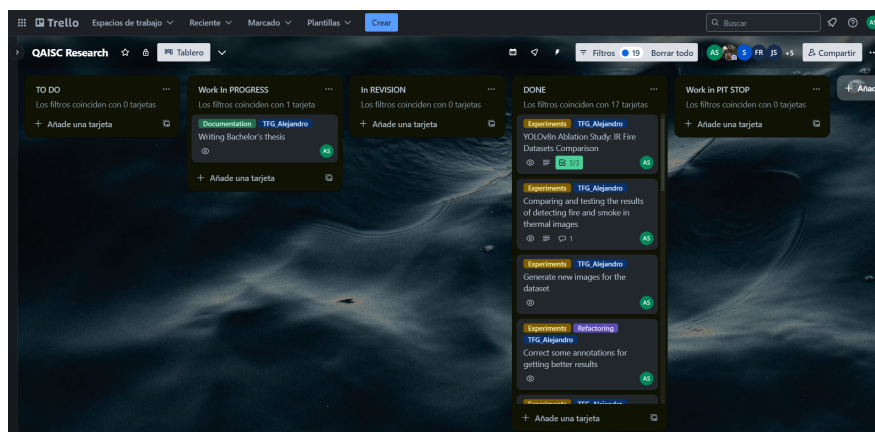


Ilustración 3.1: Tablero Kanban en Trello utilizado para la gestión de tareas en el desarrollo del Trabajo Fin de Grado

El uso de Trello como tablero Kanban no solo facilitó la organización del trabajo, sino que también proporcionó una herramienta visual eficaz para mantener el seguimiento de las tareas y asegurar la colaboración fluida con los tutores [50]. Al visualizar las tareas pendientes, en proceso y completadas, se mejoró la capacidad de priorizar y ajustar el enfoque en función de los resultados obtenidos.

De esta forma, la combinación de una metodología basada en objetivos y la implementación de *Kanban* con *Trello* proporcionó un marco eficiente para gestionar el trabajo en este TFG. El enfoque permitió una planificación estructurada y flexible, facilitó la colaboración y optimizó la gestión del tiempo y los recursos a lo largo del proyecto [26].

Capítulo 4

Herramientas y Tecnologías

Durante el desarrollo se empleó una variedad de herramientas de software y tecnologías que resultaron fundamentales para alcanzar los objetivos del proyecto. Estas herramientas no solo ayudaron a resolver los desafíos técnicos encontrados, sino que también proporcionaron una base sólida para el desarrollo eficiente y la implementación de las soluciones planteadas. A continuación, se describen las principales tecnologías y herramientas utilizadas, clasificadas según su función dentro del proyecto.

4.1. Lenguajes de Programación y Bibliotecas

- **Python:** este lenguaje fue la base de todo el desarrollo del proyecto debido a su versatilidad, simplicidad y su amplio ecosistema de bibliotecas dedicadas a la visión por computador e inteligencia artificial. Python facilitó la implementación rápida de algoritmos complejos y proporcionó una plataforma flexible para el manejo de datos y la experimentación con diferentes modelos.
- **NumPy:** como biblioteca clave para el cálculo numérico en Python, NumPy fue esencial para la manipulación eficiente de matrices y arrays, permitiendo realizar operaciones matemáticas complejas con grandes volúmenes de datos. Su capacidad para gestionar datos de forma rápida y eficiente fue vital en el preprocesamiento de la información.
- **OpenCV:** la biblioteca de visión por computador OpenCV (*Open Source Computer Vision Library*) fue crucial para las tareas de procesamiento de imágenes y vídeo. Con una amplia gama de funciones para la manipulación, análisis y mejora de imágenes, esta biblioteca permitió realizar operaciones como la detección de objetos y la transformación de imágenes, que fueron pilares fundamentales en el proyecto de visión por computador.
- **Matplotlib:** fue empleada para la visualización de los resultados y el análisis de los datos. Esta biblioteca fue indispensable para crear gráficos y representaciones visuales

claras de los resultados obtenidos, lo que facilitó la interpretación de los datos y la presentación de conclusiones de manera comprensible.

4.2. Frameworks de Aprendizaje Profundo

- **PyTorch**: este fue el framework principal utilizado para el desarrollo y entrenamiento de redes neuronales. Creado por Facebook's AI Research lab, PyTorch destaca por su flexibilidad y su capacidad de realizar cálculos eficientes en GPUs. Este framework permite una rápida iteración y ajuste de modelos complejos, siendo crucial en las fases de desarrollo e investigación del proyecto. Además, su estructura intuitiva de gráficos dinámicos facilita la depuración y el desarrollo experimental.
- **TensorFlow**: aunque PyTorch fue el framework principal, TensorFlow también jugó un papel importante en tareas específicas, como el despliegue de modelos y algunas operaciones de procesamiento de datos. TensorFlow, desarrollado por Google Brain, se utilizó especialmente debido a su robustez en tareas de clasificación y segmentación de incendios, basado en investigaciones como [46], donde fue empleado en la detección de incendios en imágenes aéreas. La integración de TensorFlow en ciertos aspectos del proyecto permitió complementar el trabajo realizado en PyTorch y facilitar la compatibilidad con diversas herramientas y plataformas.

4.3. Herramientas de Desarrollo y Control de Versiones

- **Visual Studio Code**: pese a que técnicamente es un editor de código fuente, Visual Studio Code se utilizó como el entorno de desarrollo principal debido a su extensibilidad y potentes características que lo acercan a un IDE completo [33]. Su capacidad para integrar numerosas extensiones mejoró significativamente el flujo de trabajo, permitiendo la depuración, el control de versiones integrado y el soporte para múltiples lenguajes de programación. La facilidad de personalización del entorno hizo que esta fuera una herramienta eficiente para el desarrollo y depuración del código.
- **GitHub y Git**: para el control de versiones y la colaboración en el proyecto, se utilizaron Git y GitHub. Estas herramientas fueron fundamentales para gestionar los cambios en el código de manera organizada, mantener un historial detallado del progreso, y facilitar la revisión.

4.4. Herramientas de Contenerización y Despliegue

- **Docker**: fue empleado para la contenerización de las aplicaciones desarrolladas, garantizando la consistencia del entorno de ejecución independientemente del sistema en el que se desplegaran. Esta herramienta fue clave para facilitar la portabilidad de las

aplicaciones y asegurar que los modelos funcionaran de la misma manera en distintos entornos como desarrollo y producción.

4.5. Herramientas de Anotación y Generación de Datos

- **Computer Vision Annotation Tool (CVAT):** CVAT se utilizó como la herramienta principal para la anotación de imágenes y vídeos, una tarea fundamental en la preparación de datos para el entrenamiento de modelos de visión por computador. CVAT proporciona una interfaz web intuitiva que soporta varios tipos de anotación, tales como la detección de objetos (con formatos como YOLO y COCO), la segmentación semántica y el seguimiento de objetos [37]. Su capacidad para manejar diferentes formatos de salida, como el formato YOLO (usado para la detección de objetos con cajas delimitadoras), facilitó la creación y gestión de grandes cantidades de datos anotados con precisión.
- **Getimg.ai:** Para aumentar el conjunto de datos de entrenamiento, se empleó Getimg.ai. Esta herramienta de generación y aumento de datos fue clave para mejorar la robustez del modelo, permitiendo ampliar la cantidad y variedad de datos sin necesidad de capturar imágenes adicionales, lo cual fue crucial para mejorar el rendimiento de los modelos en escenarios más diversos.

4.6. Hardware Especializado

- **Cámaras Pan-Tilt-Zoom (PTZ):** las cámaras PTZ fueron el componente de hardware utilizado para la captura de imágenes y videos en el proyecto. Su capacidad de mover y ajustar el zoom de manera remota proporcionó una gran flexibilidad, permitiendo obtener imágenes desde múltiples ángulos y distancias. Esto fue especialmente útil en las tareas de detección de objetos y seguimiento, donde la capacidad de ajustarse dinámicamente a la escena fue esencial.

La combinación de estas herramientas y tecnologías no solo fue determinante para superar los desafíos técnicos que surgieron a lo largo del proyecto, sino que también permitió optimizar el flujo de trabajo y asegurar la calidad y precisión de los resultados obtenidos. Cada una de estas herramientas jugó un papel crucial en el éxito del proyecto, desde el desarrollo del código hasta la implementación y validación de los modelos de aprendizaje profundo y visión por computador.

Capítulo 5

Análisis y Desarrollo del proyecto

5.1. Análisis del Funcionamiento de la Cámara PTZ y Planificación

El desarrollo del proyecto comenzó con la evaluación del equipo proporcionado, en este caso la cámara PTZ Biespectro térmico-óptica de Hikvision, con el objetivo de utilizar su sensor térmico para la detección de anomalías en áreas portuarias y costeras. Esta elección se fundamentó en la importancia de detectar con rapidez posibles incendios o anomalías térmicas, donde la capacidad de identificación temprana es crucial. En situaciones de baja visibilidad, como durante la noche o en condiciones climáticas adversas, el sensor térmico puede ofrecer ventajas significativas frente al sensor óptico, que podría no captar información suficiente en esos escenarios.

Con esta idea en mente, lo primero que necesitábamos era entender el tipo de datos con los que íbamos a trabajar y sus características. Por ello, comenzamos analizando el funcionamiento de la cámara proporcionada, las imágenes capturadas y el entorno en el que estaba ubicada. En este contexto, se nos otorgó acceso a una cámara situada en la azotea del Centro Comercial El Mirador en Las Palmas de Gran Canaria, donde PLOCAN tiene una ubicación estratégica, en una zona con una alta actividad costera.

Como podemos observar en la Ilustración 5.1, la cámara es capaz de capturar imágenes tanto RGB como en escala térmica. En este caso, vemos la estructura de PLOCAN y un aerogenerador durante el atardecer, como ejemplos de objetos típicos del entorno donde se encuentra la cámara.



Ilustración 5.1: Imágenes capturadas por la **cámara PTZ Biespectro térmico-óptica de Hikvision**, mostrando comparaciones entre el sensor *óptico* (izquierda) y el *térmico* (derecha). La fila superior corresponde a la **Plataforma Oceánica de Canarias (PLOCAN)** y la fila inferior a un **aerogenerador**.

A partir de estas imágenes, pudimos comparar y analizar varios aspectos útiles para el desarrollo del modelo. Uno de los principales factores considerados fue la escala de color, luminosidad e intensidad en ambos modos. El sensor óptico proporciona imágenes de alta calidad en formato RGB, lo que facilita la identificación de objetos en condiciones normales, siempre que no haya factores climáticos adversos. Por otro lado, el sensor térmico destacó por su capacidad para resaltar objetos en diversos momentos del día, incluida la noche, aunque con una calidad y enfoque ligeramente inferiores. Las imágenes térmicas se presentaron en modo *Whitehot*, que utiliza una escala donde los objetos más calientes se representan en tonos más claros o blancos, mientras que los más fríos se muestran en grises o negros. Por ejemplo, un objeto que supera los 40°C se visualizaría en un tono mucho más blanco que el mar, como puede apreciarse en la ilustración, dependiendo del umbral térmico establecido.

Sin embargo, uno de los principales retos fue que no disponíamos de las imágenes en formato RAW, lo que limitaba el acceso a información más precisa, como la temperatura en cada píxel, la intensidad y otros atributos relevantes. Esta falta de acceso a datos sin procesar dificultó la interpretación precisa de las imágenes, lo que, en consecuencia, impidió realizar una aproximación más exacta en el desarrollo del modelo. A pesar de estas limitaciones, al comprender el funcionamiento de la cámara y los datos disponibles, pudimos avanzar en la planificación y definir los requisitos iniciales y las fases clave para alcanzar nuestro objetivo.

5.2. Investigación de Modelos y Arquitecturas para la Detección Térmica

Tras definir los primeros requisitos del proyecto, el siguiente paso fue realizar un estudio del estado del arte para identificar los enfoques más avanzados y relevantes en la detección de anomalías térmicas. Dado que el proyecto se centraba en el uso del sensor térmico de la cámara, comenzamos investigando papers y métodos que abordaran la detección de incendios utilizando imágenes térmicas. Aunque nuestro entorno de pruebas estaba relacionado con zonas costeras y portuarias, consideramos que lo ideal sería encontrar métodos aplicables en distintos entornos, como áreas industriales o boscosas.

Sin embargo, enfocarnos en la detección de fuegos o anomalías térmicas en escalas *infrarrojas* (IR) dificultó la búsqueda, ya que descubrimos que la mayoría de las investigaciones se centraban en la detección de incendios con imágenes en formato RGB. Esto es comprensible, dado que las imágenes RGB permiten una mejor distinción entre el fuego, el humo y otros elementos del entorno.

A pesar de esta dificultad, encontramos una solución prometedora en el trabajo titulado *Aerial Imagery Pile Burn Detection using Deep Learning: The FLAME Dataset* [46]. Aunque este estudio se enfocaba en la detección de sólo fuego, presentaba dos soluciones diferentes: por un lado, la **clasificación binaria** de imágenes (fuego/no fuego) y, por otro, la **segmentación semántica** del fuego dentro de la imagen. Este enfoque resultó atractivo para nosotros, ya que, además de obtener resultados positivos en sus pruebas, proporcionaba el código implementado que pudimos obtener a través de *Papers with Code*¹, junto con un extenso dataset llamado FLAME², ideal para entrenar, validar y probar los modelos.

El artículo de referencia, junto con el código disponible y el conjunto de datos proporcionado, lo convirtieron en el candidato perfecto para nuestra primera aproximación al problema.

5.2.1. El conjunto de datos FLAME

Como hemos mencionado anteriormente, una de las principales contribuciones del trabajo de Alireza Shamsoshora et al. es el conjunto de datos **FLAME** (*Fire Luminosity Airborne-based Machine learning Evaluation*). Dicho dataset fue desarrollado para abordar el problema de la detección automática de incendios en entornos forestales y rurales, utilizando imágenes capturadas desde UAVs (*Unmanned Aerial Vehicles*) y drones en distintas modalidades, tanto en formato RGB como escalas térmicas. Además, FLAME es un recurso valioso y que ha sido empleado en otras investigaciones relacionadas con la detección de incendios debido a su diversidad de datos.

¹Disponible en: <https://paperswithcode.com/paper/aerial-imagery-pile-burn-detection-using-deep>

²Disponible en: <https://iee-dataport.org/open-access/flame-dataset-aerial-imagery-pile-burn-detection-using-drones-uavs>

Características del dataset

El conjunto de datos **FLAME** incluye una amplia variedad de información recopilada mediante diferentes tecnologías y dispositivos, específicamente diseñados para la detección de incendios en entornos naturales. Las imágenes fueron capturadas desde UAVs y drones, utilizando cámaras de diversos tipos, lo que permite un análisis exhaustivo y preciso de los focos de fuego.

Entre los dispositivos empleados se encuentran tanto cámaras RGB estándar como cámaras térmicas, proporcionando así una perspectiva dual del fenómeno del incendio. Por ejemplo, se utilizaron cámaras como la **Zenmuse X4S** para grabar vídeos en formato RGB, con una resolución de **1280x720 píxeles** y una tasa de **29 FPS** (fotogramas por segundo). Además, la cámara térmica **FLIR Vue Pro R** registró imágenes en diferentes modos de captura térmica, como *WhiteHot* y *GreenHot*, con una resolución de **640x512 píxeles** y una tasa de **30 FPS**. Estas capturas térmicas son fundamentales para identificar focos de fuego que no son visibles en las imágenes ópticas convencionales (ver Ilustración 5.2).

El dataset cuenta con aproximadamente **39,375 fotogramas** etiquetados como “Fuego” o “No Fuego” para entrenar modelos de clasificación, junto con **8,617 fotogramas** adicionales destinados a las pruebas. Para la tarea de segmentación de incendios, se incluyen **2,003 imágenes** acompañadas de sus correspondientes máscaras de anotación con precisión a nivel de píxel. Estas máscaras son esenciales para entrenar algoritmos capaces de segmentar de manera precisa las áreas afectadas por el fuego, una tarea clave en la segmentación semántica.

Las imágenes proporcionan una cobertura completa de diferentes etapas del incendio, desde el inicio hasta la extinción, lo que aporta una diversidad temporal significativa. Esta variabilidad es crucial para desarrollar modelos de aprendizaje automático que puedan adaptarse a situaciones reales en diversos escenarios.

En cuanto a los dispositivos utilizados, se emplearon drones como el **DJI Phantom 3**, que capturó vídeos en alta resolución (**3840x2160 píxeles**) para los conjuntos de datos de prueba. La combinación de estos drones y cámaras avanzadas ofrece una gran flexibilidad para registrar los incendios desde distintas alturas y ángulos, lo que incrementa la calidad y aplicabilidad de los datos en estudios posteriores.

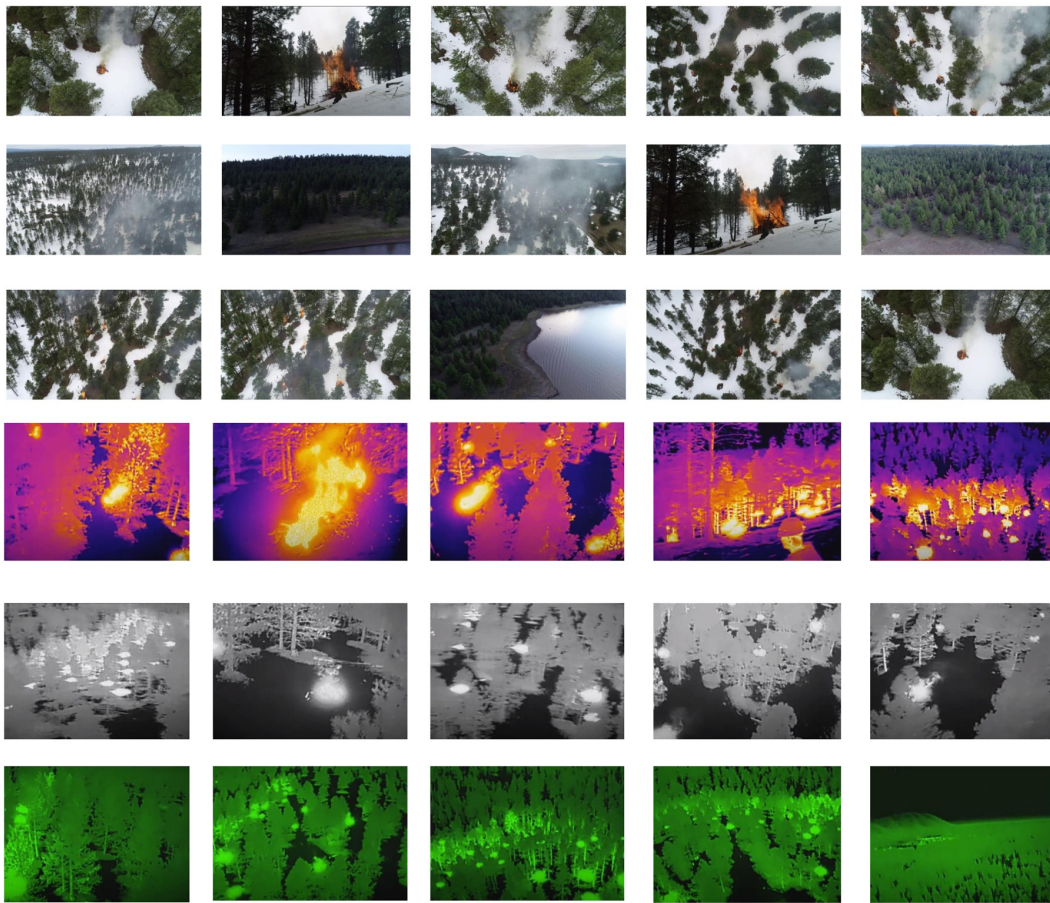


Ilustración 5.2: Comparación de imágenes RGB y térmicas en diferentes modalidades capturadas en el dataset FLAME. La imagen muestra capturas desde UAVs y drones, utilizando cámaras en formato RGB y térmico (*WhiteHot*, *Fusion*, y *GreenHot*), que permiten la detección precisa de incendios y su evolución. Las primeras filas representan imágenes ópticas, mientras que las últimas tres muestran diferentes formas de visualización térmica del fuego y el entorno circundante. Fuente: Adaptado de *FLAME dataset: Aerial imagery pile burn detection using drones (UAVs)* por Alireza Shamsoshoara et al. (2021), *IEEE Dataport*.

5.2.2. Clasificación de imágenes de incendios con Xception

Una vez revisados los datos proporcionados, decidimos probar el código disponible en GitHub que emplea el modelo **Xception** para la clasificación de imágenes aéreas, dividiéndolas en dos categorías: “Fuego” y “No Fuego”. Aunque el repositorio no ha sido actualizado desde 2021, consideramos que representaba una buena base para comprender cómo funciona el modelo y adaptarlo a nuestros experimentos.

Para ejecutar el modelo, configuramos un entorno Docker con soporte para CUDA, lo que nos permitió aprovechar la aceleración por GPU. Las bibliotecas principales que utilizamos

fueron:

- ✓ **TensorFlow y Keras**: La infraestructura básica sobre la que se construye y entrena el modelo. Keras, con su API sencilla, facilitó la definición de la arquitectura Xception y su posterior entrenamiento.
- ✓ **NumPy**: Fundamental para manipular y organizar los datos de entrada, permitiendo gestionar de manera eficiente los conjuntos de imágenes y etiquetas.
- ✓ **OpenCV**: Esencial para la carga y preprocesamiento de imágenes, incluyendo las capturadas en formatos variados.
- ✓ **Matplotlib**: Utilizado para la visualización de resultados, permitiéndonos observar cómo el modelo procesa los datos y los resultados de predicción.

La arquitectura del modelo Xception que empleamos (ver Figura 5.3) se basa en convoluciones separables en profundidad, que optimizan el uso de recursos al reducir significativamente el número de parámetros, sin perder precisión. Esta característica es crucial cuando se trabaja con imágenes de alta resolución tomadas desde drones, como en nuestro caso. El proceso comienza con una **capa de entrada** que recibe imágenes de dimensiones $254 \times 254 \times 3$, las cuales son normalizadas dividiendo sus valores entre 255, como se observa en el bloque de preprocesamiento.

El modelo emplea varias capas convolucionales profundas. En las primeras capas, que incluyen operaciones de convolución 2D con filtros 3×3 y un *stride* de 2×2 , se capturan características simples como bordes y texturas. Más adelante, a medida que se profundiza en la red, las capas convolucionales separables en profundidad extraen patrones más complejos, como la presencia de llamas o el entorno del paisaje. Este diseño, mostrado en la Figura 5.3, incluye bloques repetidos de normalización por lotes y activación **ReLU**, lo que ayuda a estabilizar el entrenamiento y mejorar la eficiencia del modelo.

Una de las características destacadas de esta arquitectura es la inclusión de **conexiones de atajo** (shortcuts), similares a las que utiliza ResNet, lo que permite que la información fluya de manera más eficiente a través de la red, ayudando a mitigar problemas como la desaparición del gradiente en redes profundas.

Finalmente, el modelo culmina con una **capa densa** de dos unidades, que corresponde a la clasificación binaria de las imágenes en “Fuego” y “No Fuego”. La función de activación utilizada en la salida es *sigmoide*, lo que permite interpretar los resultados como probabilidades, asignando un valor entre 0 y 1 según la probabilidad de que la imagen contenga fuego.

Este diseño está representado en la Figura 5.3, donde se puede observar el flujo de datos desde la capa de entrada hasta la salida, pasando por varias etapas de convoluciones y funciones de activación.

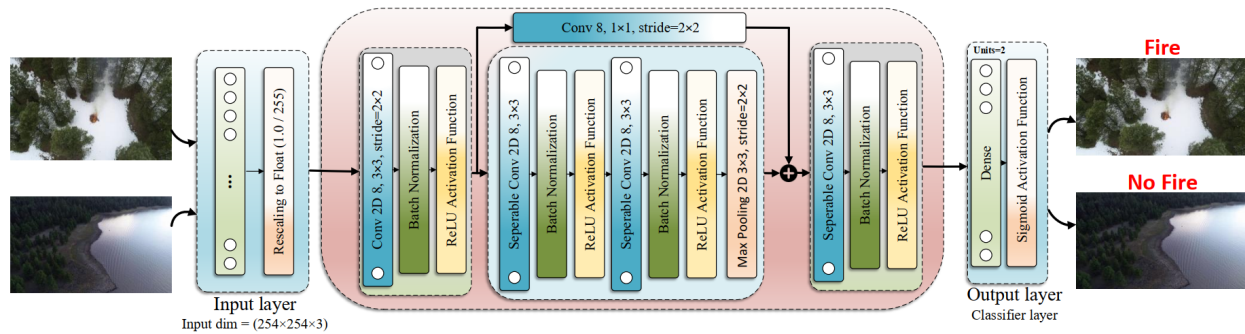


Ilustración 5.3: Esquema de la arquitectura del modelo Xception utilizado para la clasificación de imágenes de incendios en “Fuego” y “No Fuego”. Fuente: Adaptado del repositorio de GitHub *Fire Detection UAV Aerial Image Classification Segmentation* por Alireza Shamsoshoara et al. (2021) <https://github.com/AlirezaShamsoshoara/Fire-Detection-UAV-Aerial-Image-Classification-Segmentation-UnmannedAerialVehicle>.

5.2.3. Entrenamiento, validación y testing del modelo de clasificación

A la hora de desarrollar un modelo, es fundamental dividir el proceso en unas fases claves: entrenamiento, validación y testing. El **entrenamiento** es la etapa en la que el modelo aprende a partir de los datos, ajustando sus parámetros internos para minimizar el error en la predicción. La validación se utiliza durante el entrenamiento para evaluar el rendimiento del modelo en datos no vistos, ajustando hiperparámetros e intentando evitar el sobreajuste u *overfitting*, que es que el modelo se ajusta demasiado a los datos de entrenamiento dando lugar a alta precisión durante este pero luego no es capaz de generalizar con datos que no ha visto nunca. Finalmente, el **testing** es la fase en la que se evalúa el modelo final en un conjunto completamente independiente de datos, para medir su capacidad de generalización.

En este punto, lo que hicimos para nuestros experimentos, fue preparar dos conjuntos de datos a partir del dataset FLAME, con el objetivo de analizar los resultados. Para la fase de entrenamiento y validación, utilizamos **39375** frames en formato RGB obtenidos con la cámara **Zenmuse**. Por otro lado, la fase de test se realizó con **8617** frames grabados con la cámara **Phantom**.

El propósito principal de esta primera prueba era familiarizarnos con el funcionamiento del código, entender cómo se procesaban las imágenes, de qué manera se hacía uso de hiperparámetros y los resultados que se obtenían. Aunque si bien estábamos trabajando con imágenes RGB, como mencionamos anteriormente, el objetivo final era probar el modelo utilizando imágenes en escala térmica que el dataset FLAME también ofrece.

Para ello, se realizaron pruebas adicionales con un conjunto mucho más reducido de **852** imágenes térmicas, extraídas de un vídeo de 89 segundos sin etiquetar, por lo que tuvimos que hacerlo nosotros manualmente y que finalmente nos quedamos con **778**. Si bien este conjunto es considerablemente menor en comparación con el de las imágenes RGB, pudimos hacer uso de él con fines de experimentación y evaluación del modelo en este tipo de imágenes.

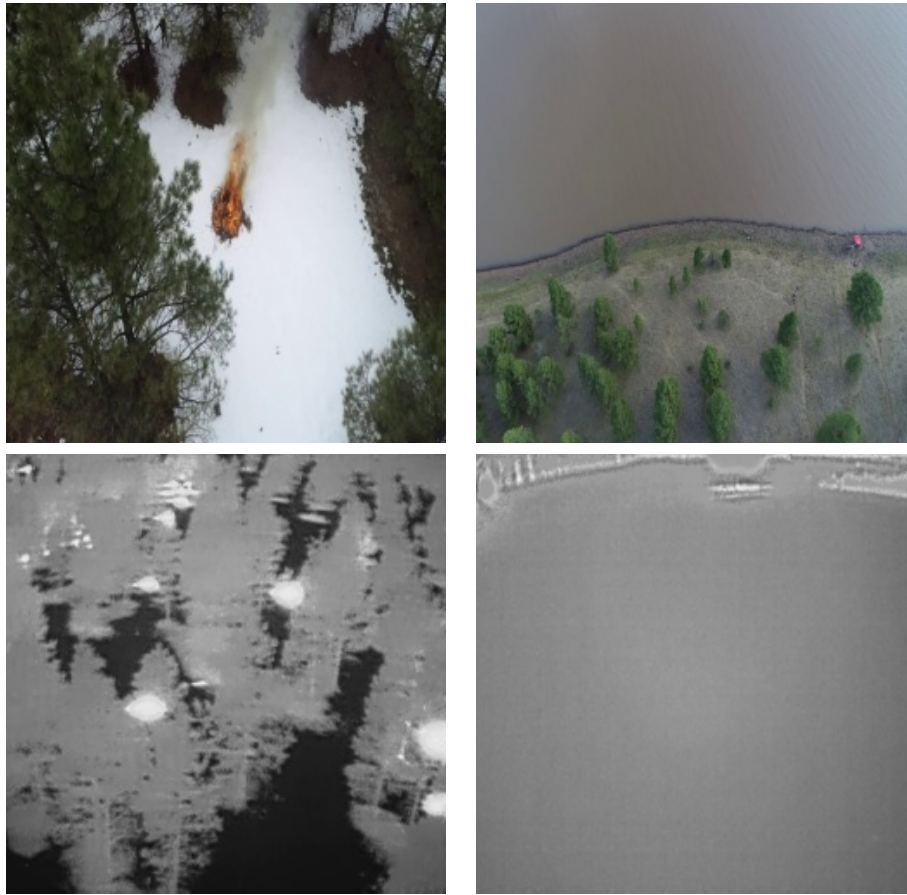


Ilustración 5.4: Imágenes obtenidas del dataset de FLAME. La columna de la izquierda representan la clase “Fuego” y la derecha “No fuego”, siendo la fila superior en formato RGB y la inferior IR en escala *Whitehot*.

Fase de entrenamiento y validación

Durante esta fase, en el código se realizan ciertas funciones y pasos que son claves para el correcto ajuste del modelo, así como de una adecuada gestión de los hiperparámetros.

Como configuración básica, el modelo se entrenó utilizando un tamaño de lote (*batch size*) de **32** y se ejecutó durante **40** épocas. Una época se refiere a pasar por completo el conjunto de datos de entrenamiento una vez, mientras que el tamaño de lote define cuántos ejemplos se procesan antes de actualizar los parámetros del modelo. Para mejorar la diversidad del conjunto de datos y prevenir el sobreajuste, se utilizó *data augmentation*, que es una técnica que genera nuevas imágenes de entrenamiento de manera artificial a partir del conjunto existente. Esto se logra mediante transformaciones como rotaciones, volteos y escalados, aumentando el número de muestras disponibles sin necesidad de recolectar más datos. En Keras, esta técnica se implementa fácilmente con la clase **Sequential** y funciones como **RandomFlip(“horizontal”)**, que invierte aleatoriamente las imágenes de forma horizontal, y **RandomRotation(0.1)**, que aplica rotaciones aleatorias de hasta un 10 %.

Un aspecto crucial para entrenar modelos con conjunto de datos posiblemente desequilibrados es la asignación de pesos a las clases. En el caso del código de Alireza Shamsoshoara, las clases “Fuego” y “No fuego” no obtienen la misma cantidad de ejemplos, por lo que se asigna un peso a cada clase basado en su frecuencia. Esto asegura que el modelo no favorezca la clase mayoritaria y logre un rendimiento equilibrado entre ambas.

Además, la división del conjunto de datos en subconjuntos para entrenamiento y validación es fundamental. Se usó la función `image_dataset_from_directory` de Keras, que permite cargar imágenes directamente desde un directorio y dividir las en subconjuntos de entrenamiento y validación. Se define una proporción de **80 %** para el entrenamiento y **20 %** para la validación, y especificamos una semilla (*seed*) para garantizar que no haya solapamientos entre los conjuntos.

El optimizador elegido para este problema fue **Adam** (*Adaptive Moment Estimation*), un método ampliamente utilizado y efectivo para problemas de clasificación. Adam combina lo mejor de dos enfoques populares: el descenso por gradiente con momento y el algoritmo **RMSProp** (*Root Mean Square Propagation*). El momento ayuda a acelerar la convergencia del modelo, mientras que RMSProp ajusta dinámicamente la tasa de aprendizaje para cada parámetro, lo que mejora la estabilidad y la eficiencia del entrenamiento.

La tasa de aprendizaje (*learning rate*) seleccionada fue de **1e-3**. Este valor define el tamaño de los pasos que da el modelo al ajustar los pesos después de cada iteración. Un valor demasiado grande puede llevar al modelo a saltarse el punto óptimo de convergencia, mientras que uno demasiado pequeño podría hacer que el modelo tarde demasiado en ajustarse o quede atrapado en un mínimo local. Un valor de **1e-3** es considerado estándar, equilibrando la velocidad y la precisión del ajuste de pesos.

Al ser un problema de clasificación binaria, la función de pérdida que se empleó en el código fue **binary_crossentropy**. Esta función mide la diferencia entre las predicciones del modelo y las etiquetas verdaderas (0 o 1), y se considera la mejor opción para problemas de clasificación con dos clases, ya que calcula la entropía cruzada, que es una medida de la incertidumbre entre las predicciones del modelo y los valores reales.

En conjunto, todas estas técnicas y configuraciones contribuyen a mejorar la calidad del entrenamiento, buscando evitar el sobreajuste mediante la variación de los datos de entrada, una buena separación entre los conjuntos de entrenamiento y validación, y mitigando el sesgo causado por el desbalance de clases gracias a la asignación de pesos apropiados.

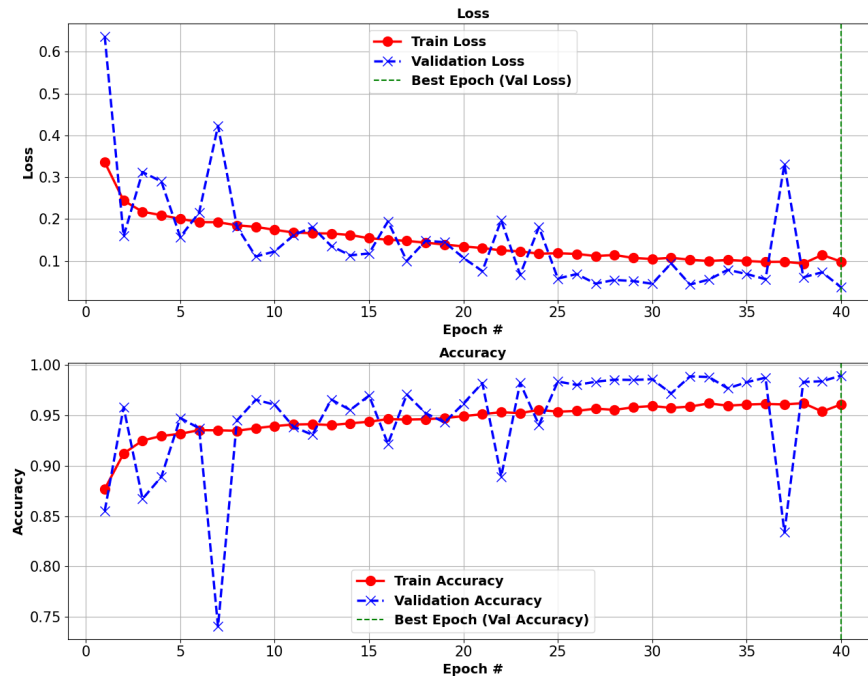


Ilustración 5.5: Gráfica de rendimiento del modelo de clasificación binaria Xception durante 40 épocas en el conjunto de datos **RGB** de FLAME, validando pérdida y precisión.

La gráfica de la Figura 5.5 muestra el rendimiento del modelo Xception entrenado para clasificación binaria sobre el conjunto de datos RGB de FLAME a lo largo de 40 épocas, usando un tamaño de lote de 32. Observamos un comportamiento positivo en las métricas de entrenamiento: la pérdida sigue un descenso progresivo, indicando que el modelo mejora en su ajuste a los datos, y la precisión se eleva de manera consistente, superando incluso el 95 % en las últimas épocas.

Sin embargo, las métricas de validación revelan un comportamiento menos estable. Aunque el modelo logra buenas precisiones en varias épocas, existen fluctuaciones notables con picos y valles tanto en la pérdida como en la precisión de validación. Este patrón sugiere una posible falta de generalización, lo cual podría indicar que el modelo está ajustándose demasiado a las características específicas del conjunto de entrenamiento y, por lo tanto, tiene dificultades para mantener un rendimiento consistente en datos no vistos. Estas oscilaciones son indicativas de sobreajuste (overfitting) o, en menor medida, subajuste en ciertas épocas, donde el modelo podría no estar capturando patrones suficientemente generales para clasificar correctamente en el conjunto de validación.

A pesar de estas variaciones, decidimos evaluar el modelo Xception en las imágenes IR del conjunto de datos FLAME para comprobar si el modelo puede mejorar su capacidad de generalización en un dominio diferente, aprovechando la naturaleza de las imágenes en infrarrojo, que podrían resaltar diferentes patrones en comparación con las imágenes RGB.

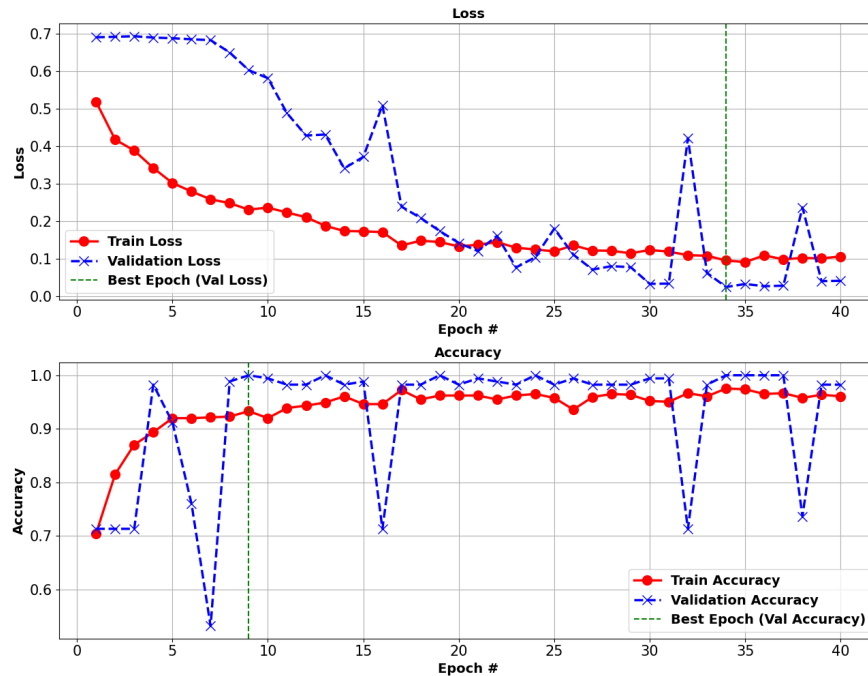


Ilustración 5.6: Gráfica de rendimiento del modelo de clasificación binaria Xception durante 40 épocas en el conjunto de datos **IR** de FLAME, validando pérdida y precisión.

En el caso del entrenamiento y validación del conjunto de incendios IR (Figura 5.6, observamos un comportamiento similar al entrenamiento anterior con imágenes RGB: la pérdida de entrenamiento desciende consistentemente, lo que es un buen indicador de que el modelo está aprendiendo a ajustar sus parámetros para la clasificación. Asimismo, la precisión en el entrenamiento aumenta y se mantiene estable en niveles altos, lo que indica que el modelo puede clasificar correctamente la mayoría de los ejemplos de entrenamiento.

No obstante, el desempeño en el conjunto de validación presenta variaciones notables. La precisión de validación experimenta fluctuaciones significativas, alcanzando el 100% en varias épocas, pero también exhibiendo caídas abruptas. Este comportamiento sugiere que el modelo podría estar sobreajustándose a los datos de entrenamiento, lo que se refleja en la inestabilidad de la precisión en el conjunto de validación. Además, la pérdida de validación permanece elevada durante las primeras 17 épocas y disminuye de forma irregular en lugar de seguir el mismo patrón estable observado en el entrenamiento. Este desajuste entre el entrenamiento y la validación podría indicar que el modelo tiene dificultades para generalizar en el conjunto de validación, probablemente debido a la variabilidad en las imágenes IR junto a que la cantidad de datos para entrenar y validar no es muy elevada.

Fase de testing

En esta fase, se utilizó el conjunto de test para evaluar el rendimiento del modelo (usando la función **evaluate** de Keras) y ver cómo se comportaba con datos que no había visto

previamente. A través de este proceso, se obtuvieron diversas métricas, como la precisión y la pérdida, así como la **matriz de confusión**.

La matriz de confusión es una herramienta que permite evaluar el rendimiento de un modelo de clasificación. Se suele representar como una tabla que muestra el número de predicciones correctas e incorrectas hechas por el modelo, comparando las predicciones con las etiquetas reales, organizándose en cuatro categorías:

1. **Verdaderos positivos (TP)**: Son los casos en los que el modelo predice correctamente la clase “Fuego”.
2. **Falsos positivos (FP)**: Son las predicciones incorrectas donde el modelo predice “Fuego” cuando en realidad es “No Fuego”.
3. **Verdaderos negativos (TN)**: Son los casos en los que el modelo predice correctamente la clase “No Fuego”.
4. **Falsos negativos (FN)**: Son las predicciones incorrectas donde el modelo predice “No Fuego” cuando en realidad es “Fuego”.

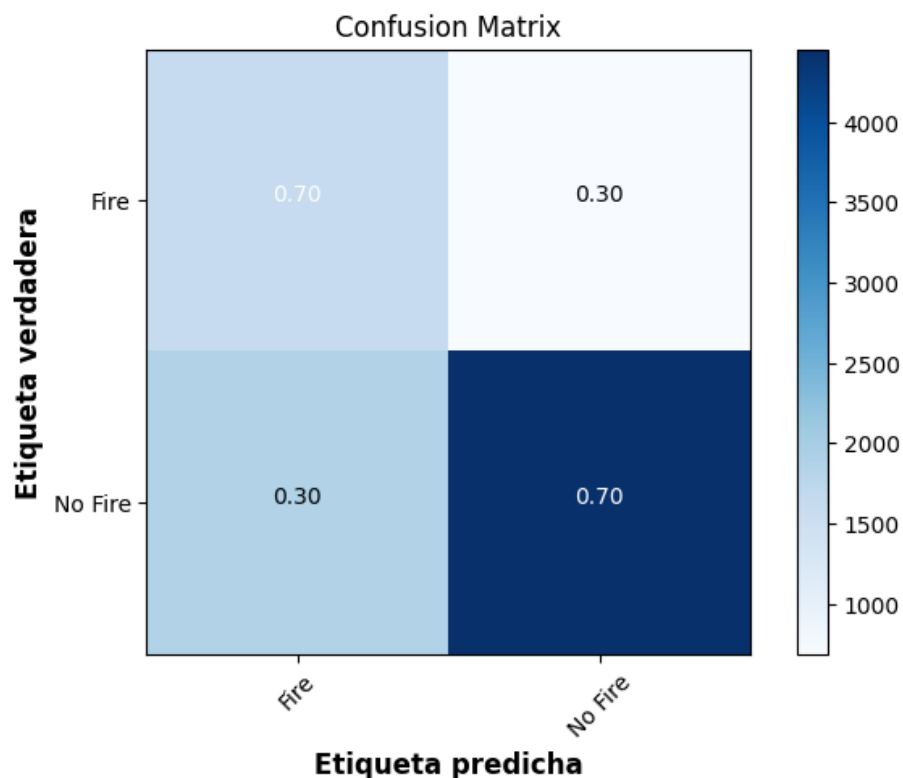


Ilustración 5.7: Matriz de confusión normalizada del modelo Xception entrenado en el conjunto **RGB** para la clasificación de fuego/no-fuego en el conjunto de prueba.

La matriz de confusión normalizada obtenida con el modelo Xception entrenado en el conjunto RGB (Figura 5.7) muestra un desempeño moderado en la clasificación de imágenes

en las categorías “Fuego” y “No Fuego”. El modelo alcanza un 70 % de precisión en ambas clases, indicando que es capaz de clasificar correctamente el 70 % de las imágenes, ya sea que contengan fuego o no. Sin embargo, también muestra un índice de error del 30 % en ambas categorías: un 30 % de las imágenes que realmente contienen fuego se clasifican incorrectamente como “No Fuego” (falsos negativos), mientras que un 30 % de las imágenes sin fuego se etiquetan erróneamente como “Fuego” (falsos positivos). Esta tasa de error sugiere que el modelo podría generar un número considerable de falsas alarmas en un contexto real, o peor, podría fallar al detectar situaciones críticas de fuego.

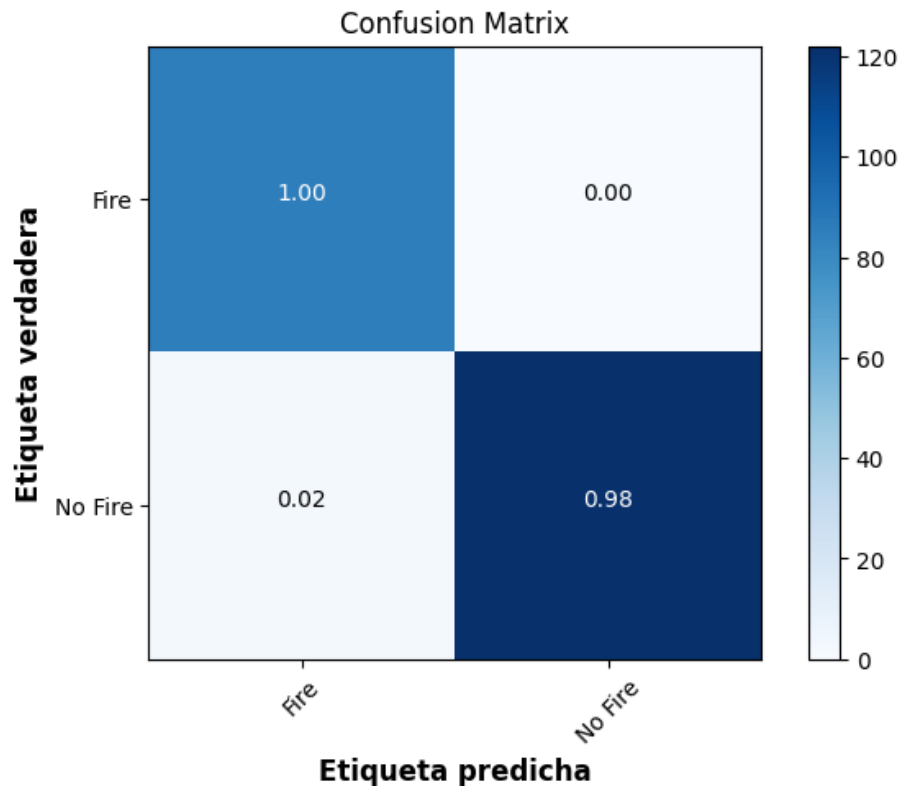


Ilustración 5.8: Matriz de confusión normalizada del modelo Xception entrenado en el conjunto **IR** para la clasificación de fuego/no-fuego en el conjunto de prueba.

Al evaluar el modelo Xception entrenado con el conjunto de imágenes infrarrojas (IR) de FLAME, los resultados reflejados en la Figura 5.8 presentan una precisión casi perfecta, mostrando una detección del 100 % tanto para la categoría de “Fuego” como de “No Fuego”. Aunque estos resultados parecen ideales a primera vista, es importante interpretar estos valores con cautela. Este nivel de precisión inusualmente alto indica un probable *sobreajuste* (overfitting) debido a la falta de variabilidad en el conjunto de datos de entrenamiento IR, lo cual lleva al modelo a aprender patrones específicos de las imágenes de entrenamiento en lugar de generalizar correctamente a nuevas imágenes.

Este sesgo hacia el conjunto de entrenamiento implica que, aunque el modelo muestra un rendimiento casi perfecto en el test de datos IR, es probable que falle al intentar clasificar

imágenes en otros contextos o escenarios diferentes. De hecho, en pruebas adicionales con imágenes IR de entornos distintos, el modelo demostró dificultades para identificar correctamente las anomalías térmicas, fallando en varias detecciones clave.

En conclusión, aunque estas pruebas iniciales de clasificación utilizando el conjunto FLAME no fueron completamente satisfactorias para nuestro objetivo final, proporcionaron una visión valiosa sobre las limitaciones del modelo y los desafíos del dataset. A partir de estos resultados, pudimos identificar la importancia de un conjunto de datos más diverso y robusto para evitar el sobreajuste, lo cual constituye un punto de partida útil para optimizar el modelo de detección de anomalías térmicas en futuras iteraciones.

5.2.4. Segmentación de imágenes de incendios con U-Net

Otra técnica propuesta para la detección de incendios es la segmentación semántica, la cual permite identificar los píxeles exactos correspondientes a las áreas afectadas por el fuego y generar una máscara que delimite estas zonas con precisión. Para llevar a cabo esta tarea, se utilizó el modelo **U-Net** [42], una arquitectura de redes neuronales convolucionales diseñada especialmente para tareas de segmentación (Figura 5.9).

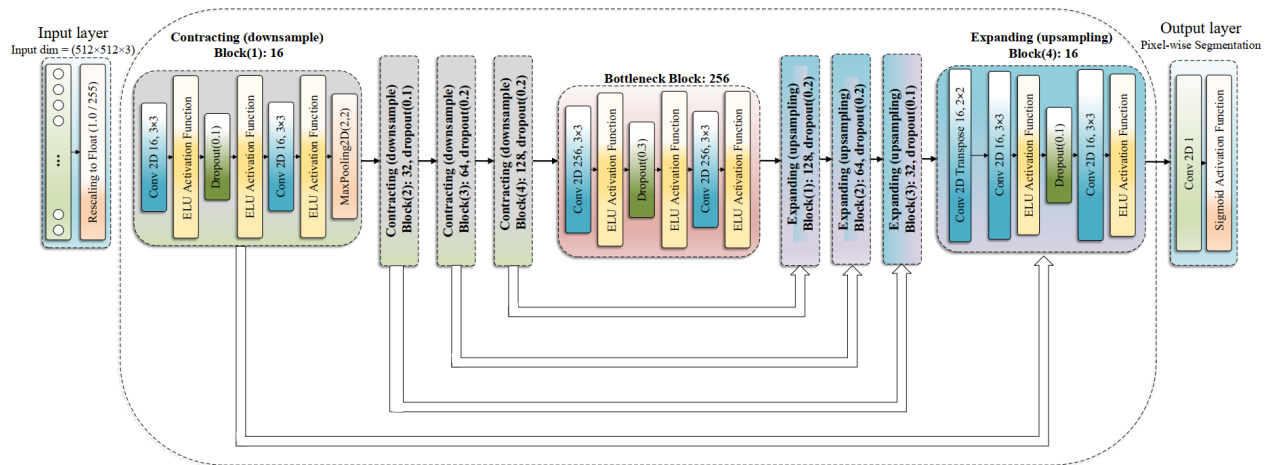


Ilustración 5.9: Esquema de la arquitectura del modelo U-Net utilizado para la segmentación de imágenes de incendios. Fuente: Adaptado del repositorio de GitHub *Fire Detection UAV Aerial Image Classification Segmentation* por Alireza Shamsoshoara et al. (2021) <https://github.com/AlirezaShamsoshoara/Fire-Detection-UAV-Aerial-Image-Classification-Segmentation-UnmannedAerialVehicle>.

El modelo U-Net comienza con una **capa de entrada**, que recibe imágenes del tamaño $521 \times 512 \times 3$ (ancho, alto y tres canales de color RGB). Se realiza una normalización, que escala los valores de los píxeles entre 0 y 1 dividiéndolos entre 255. Esta normalización se lleva a cabo mediante una función **Lambda** en el código.

La arquitectura de U-Net sigue un patrón característico, con dos caminos principales: el **camino de contracción** (o downsampling) y el **camino de expansión** (o upsampling).

En el camino de contracción, las imágenes se procesan a través de capas convolucionales sucesivas (representadas por **Conv2D**) que reducen progresivamente la resolución, mientras que las operaciones de **MaxPooling2D** extraen las características clave. Este proceso de contracción permite que el modelo capture patrones de alto nivel, como bordes y texturas, a medida que la imagen se vuelve más pequeña y sus características más abstractas. En el código, esto se realiza en bloques de capas con convoluciones 3×3 seguidas de funciones de activación **ELU** (*Exponential Linear Unit*) y operaciones de **MaxPooling** con una ventana de 2×2 .

El camino de expansión invierte el proceso de contracción, utilizando operaciones de **Conv2DTranspose** para aumentar la resolución de las características comprimidas. Cada bloque de expansión se conecta con el bloque correspondiente del camino de contracción mediante **conexiones de atajo** (skip connections). Estas conexiones permiten que la información detallada a nivel de píxel, capturada en las primeras capas, se combine con las características de alto nivel, lo que resulta en una segmentación más precisa. En el código, cada capa de **upsampling** se concatena con la función **concatenate** con su contraparte del camino de contracción para asegurar que los detalles finos no se pierdan.

Finalmente, termina con una **capa de salida** que utiliza una operación **Conv2D** con un filtro de 1×1 para reducir el número de canales a uno solo, correspondiente a la máscara binaria de la segmentación. La activación utilizada es la función **sigmoide**, que genera valores entre 0 y 1 para cada píxel, representando la probabilidad de que un píxel pertenezca a la clase “fuego” o “no fuego”.

El entrenamiento del modelo se realiza utilizando el optimizador **Adam**, conocido por su efectividad en problemas de optimización no lineal. En este caso, se utilizó una función de pérdida **binary crossentropy**, que evalúa la diferencia entre la máscara predicha por el modelo y la máscara real, ideal para problemas de segmentación binaria. Además, durante el entrenamiento, se utilizó un conjunto de métricas que incluyen **IoU** (*Intersection over Union*), **precision** y **recall**, las cuales son clave para evaluar la calidad de las máscaras generadas.

Entrenamiento y resultados

FLAME ofrecía un conjunto de **2003** frames, de los cuales el **85 %** (1702 imágenes) se utilizó para el entrenamiento, mientras que el **15 %** restante (301 imágenes) se destinó para la validación del modelo. Cada uno de estos frames estaba acompañado de su correspondiente máscara binaria, que fue generada utilizando MATLAB para etiquetar las áreas afectadas por el fuego (Figura 5.10).



Ilustración 5.10: Comparación de una imagen RGB capturada por la cámara **DJI Phantom 3** del dataset FLAME y su respectiva máscara binaria etiquetada como “fuego”.

Para el entrenamiento del modelo, se utilizó un tamaño de lote de **16** y se entrenó durante **30 épocas**. Sin embargo, con el objetivo de prevenir el sobreajuste (*overfitting*), se implementó un mecanismo de **EarlyStopping**, que detenía el entrenamiento si no se observaba mejora en la pérdida de validación durante 5 épocas consecutivas.

A continuación, se presentan los resultados obtenidos tras el entrenamiento, incluyendo tanto las máscaras predichas como las métricas de rendimiento en las imágenes RGB del conjunto de datos FLAME.

Cuadro 5.1: Evaluación del rendimiento del modelo U-Net personalizado en el conjunto de datos de incendios en RGB.

Métrica	Rendimiento (%)
Precisión (%)	93.63
Recall (%)	82.27
AUC (%)	91.12
F1-Score (%)	87.59
Sensibilidad (%)	82.27
Especificidad (%)	99.97
IOU (%)	77.91

Como se puede observar en el Cuadro 5.1, el modelo de U-Net muestra un rendimiento sólido en la segmentación de incendios. La **precisión** es alta, alcanzando un **93.63 %**, lo que indica que la mayoría de las áreas identificadas como fuego son correctas. Sin embargo, el **recall** es algo más bajo (**82.27 %**), lo que sugiere que el modelo no detecta todas las áreas afectadas por el fuego, pero sigue siendo un buen rendimiento general.

El **AUC (91.12 %)** demuestra una excelente capacidad para distinguir entre áreas con y sin fuego. El **F1-Score (87.59 %)** refleja un buen equilibrio entre precisión y recall, destacando que el modelo maneja adecuadamente ambos tipos de errores. La **especificidad** es

notablemente alta (**99.97 %**), lo que significa que casi siempre predice correctamente cuando no hay fuego.

Finalmente, el **IoU** de **77.91 %** muestra que el modelo logra una buena coincidencia entre las áreas predichas y las reales, aunque todavía hay margen de mejora en la precisión de las máscaras generadas.

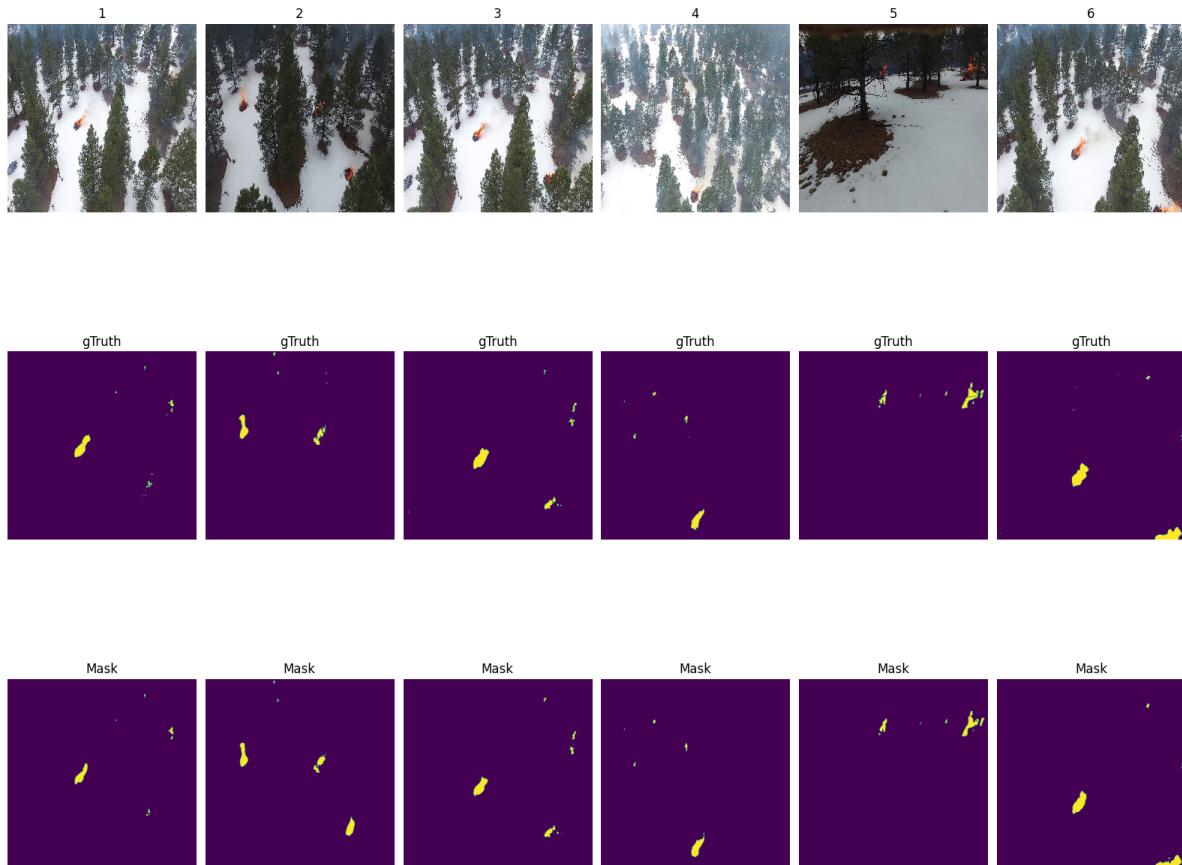


Ilustración 5.11: Imagen que muestra como primera fila imágenes RGB del dataset FLAME, seguido de las máscaras *ground truth* y las máscara predichas por el modelo U-Net.

La Figura 5.11 muestra una serie de pruebas realizadas con el modelo U-Net, donde se comparan las máscaras reales de incendios (denominadas **ground truth**) y las máscaras predichas por el modelo. Estas pruebas son fundamentales para evaluar la capacidad del modelo de segmentar correctamente las áreas afectadas por el fuego.

En la primera fila, se presentan las imágenes originales capturadas por drones. Estas imágenes corresponden a distintas escenas en las que se observan áreas con fuego, representando datos reales capturados desde el aire.

La segunda fila, etiquetada como **gTruth**, corresponde a las máscaras de referencia o **ground truth**. Estas máscaras han sido generadas manualmente o mediante un proceso de anotación, donde cada píxel amarillo indica la presencia de fuego en la imagen original. El

propósito de estas máscaras es proporcionar una referencia exacta de las áreas afectadas, contra las cuales se pueden comparar las predicciones del modelo.

La tercera fila, etiquetada como **Mask**, muestra las máscaras predichas por el modelo U-Net. Estas predicciones indican qué áreas de las imágenes originales el modelo considera que contienen fuego, basándose en el entrenamiento que recibió. Idealmente, estas máscaras deberían coincidir lo más posible con las **ground truth**, lo que indicaría un buen rendimiento del modelo.

Si observamos las imágenes generadas, podríamos decir que el modelo es capaz de segmentar correctamente las zonas de fuego en este caso de hogueras. Sin embargo, no podemos dar por hecho de que generalice correctamente debido al conjunto de datos sesgados al problema de incendios forestales, por ello hemos recopilado algunas imágenes de incendios portuarios para comprobarlo.

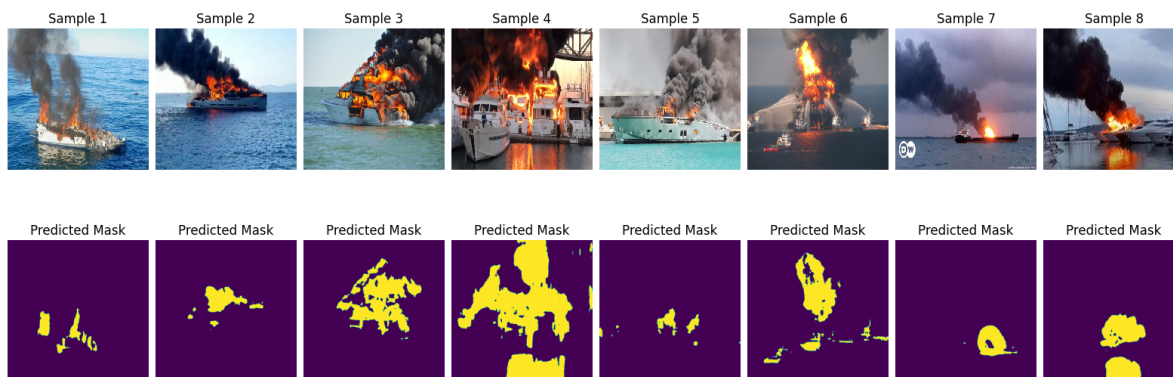


Ilustración 5.12: Resultados de las máscaras binarias predichas por el modelo de segmentación U-Net en imágenes de incendios marítimos.

Tras haber hecho la prueba con imágenes de incendios en entornos marítimos (Figura 5.12), como pueden ser fuegos en puertos, barcos o plataformas, se ve realmente que el modelo es capaz de identificar correctamente zonas de fuego, aunque con algunos errores. Zonas como el mar donde se refleja el fuego o áreas del cielo con brillos ciertamente cálidos son a veces detectadas como fuego cuando en realidad no lo son. Esto es comprensible, dado que el modelo fue entrenado con un conjunto de datos especializados en fogatas en zonas boscosas, no está preparado para detectar estos patrones que sí pueden aparecer en otros entornos. No obstante, como aún así mostró algunos resultados prometedores, planteamos entrenar el modelo con el conjunto de datos IR proporcionado por FLAME, ya que era el objetivo principal de nuestro trabajo.

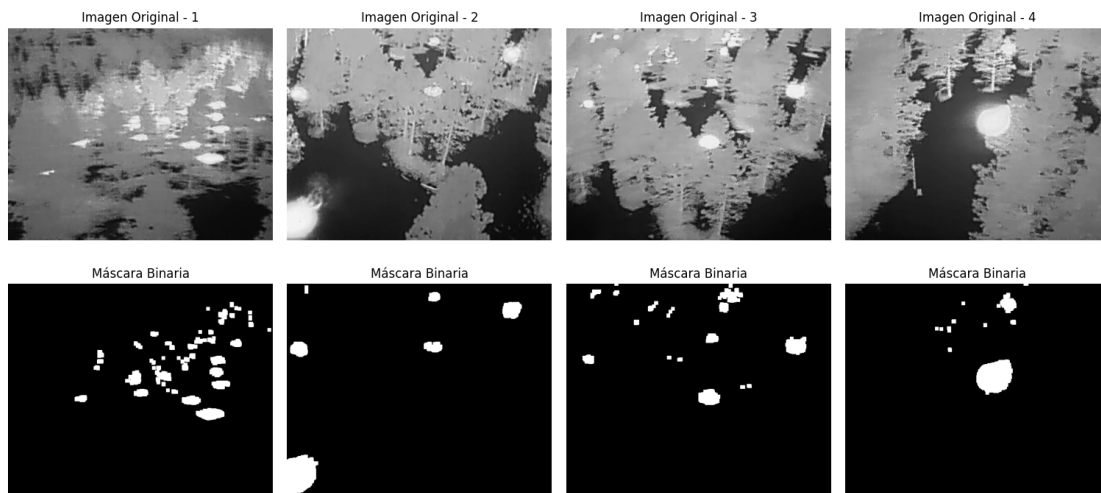


Ilustración 5.13: Comparación visual entre las imágenes originales del conjunto de datos IR de FLAME y las máscaras binarias generadas. La fila superior muestra los frames originales capturados en infrarrojo, mientras que la fila inferior presenta las máscaras binarias correspondientes, creadas utilizando técnicas de procesamiento de imagen en OpenCV. Estas máscaras resaltan las áreas de interés (zonas de fuego) mediante umbrales específicos y operaciones morfológicas, facilitando la segmentación precisa de las anomalías térmicas en las imágenes IR.

Para los frames infrarrojos (IR) del conjunto de datos FLAME, tuvimos que generar manualmente las máscaras que identifican las áreas de fuego, ya que estas no se proporcionaban en el conjunto original. Para comenzar el proceso, utilizamos la librería **OpenCV**, aprovechando sus herramientas de procesamiento de video e imágenes. Primero, extraímos los frames individuales del video capturado por el dron utilizando la clase **VideoCapture** en combinación con la función **read**, que permite leer el video y guardar cada frame como una imagen independiente.

Tras obtener los frames, determinamos el rango de valores de color característicos del fuego en las imágenes IR. Con la ayuda de un detector de color hexadecimal en línea, identificamos que las áreas de fuego en el video se encontraban, en general, dentro del rango de valores RGBA entre aproximadamente (200, 200, 200, 255) y (240, 240, 240, 255). A partir de esta información, empleamos la función **inRange** de OpenCV, que genera una máscara binaria seleccionando los píxeles dentro del rango de color especificado, marcándolos en blanco (valor 255) y dejando en negro (valor 0) aquellos fuera del umbral.

Para optimizar las máscaras y reducir el ruido presente, aplicamos operaciones morfológicas mediante la función **morphologyEx** de OpenCV. Estas operaciones ayudan a eliminar imperfecciones y a definir mejor las áreas detectadas. Primero, utilizamos la operación de apertura (**MORPH_OPEN**), que remueve pequeños elementos de ruido en el fondo, y luego aplicamos dilatación (**MORPH_DILATE**) para expandir y unificar las áreas detectadas,

mejorando así la visibilidad y precisión de las regiones de fuego en la máscara. El resultado final de este proceso se muestra en la Figura 5.13.

Cuadro 5.2: Evaluación del rendimiento del modelo U-Net personalizado en el conjunto de datos de incendios en IR.

Métrica	Rendimiento (%)
Precisión (%)	48.08
Recall (%)	84.60
AUC (%)	91.63
F1-Score (%)	61.32
Sensibilidad (%)	84.60
Especificidad (%)	98.66
IOU (%)	44.21

La tabla de rendimiento del modelo U-Net en imágenes IR del conjunto FLAME (Cuadro 5.2) muestra resultados algo limitados en comparación con el conjunto RGB. La precisión es baja (48.08 %), lo que indica una alta tasa de falsos positivos, mientras que el recall o sensibilidad es notablemente alto (84. %), reflejando que el modelo detecta la mayoría de las áreas de incendio cuando están presentes. Este desequilibrio sugiere que el modelo tiende a sobre-detectar el fuego, posiblemente debido a las características visuales menos claras en el espectro IR.

El valor de AUC es alto (91.63 %), lo que demuestra una buena capacidad para distinguir entre clases, aunque el F1-Score intermedio (61.32 %) y el IoU bajo (44.21 %) indican que la segmentación específica es menos precisa. Esto podría deberse al tamaño reducido del conjunto de datos IR y a la calidad de imagen distinta en comparación con el RGB, factores que dificultan la generalización del modelo.

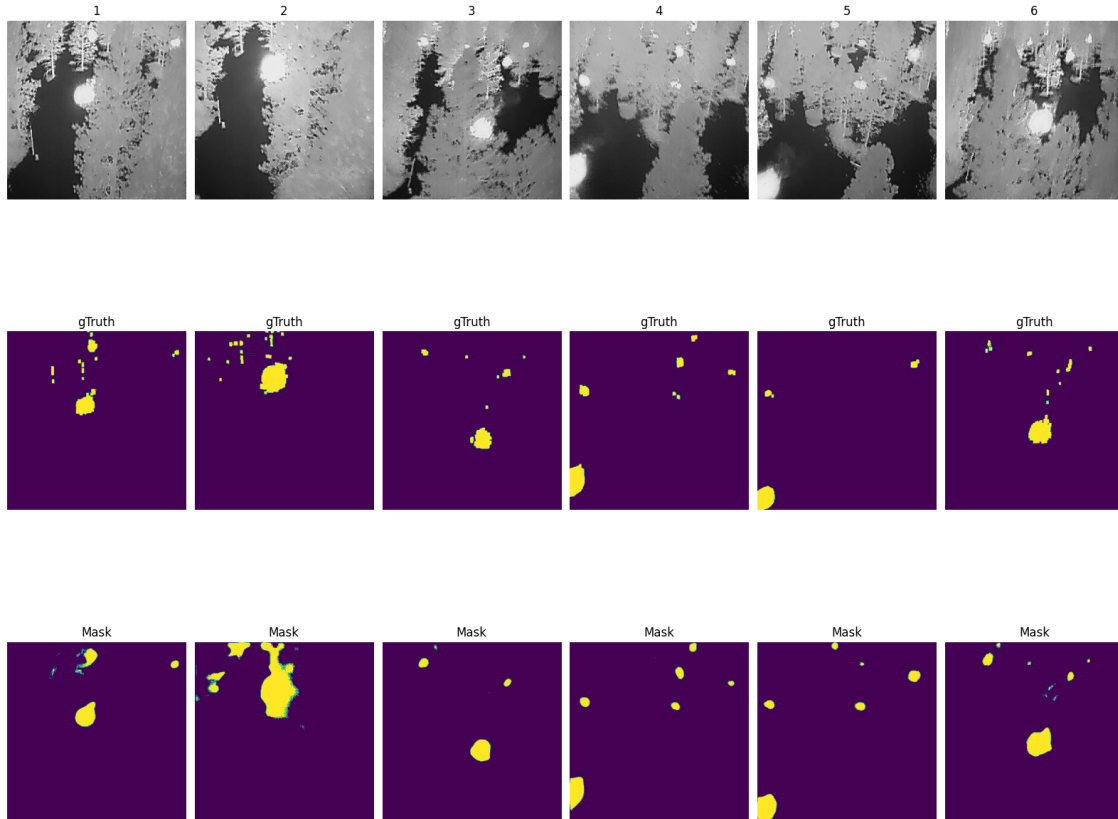


Ilustración 5.14: Imagen que muestra como primera fila imágenes IR del dataset FLAME, seguido de las máscaras *ground truth* y las máscara predichas por el modelo U-Net.

Luego, la Figura 5.14 muestra que el modelo de segmentación U-Net es capaz de identificar áreas de fuego en las imágenes capturadas por el dron, aunque suele sobreestimar algunas regiones y asignar píxeles que no representan fuego real. Para evaluar la precisión del modelo en situaciones no contempladas en el entrenamiento, también realizamos pruebas con imágenes reales de incendios y escenas sin presencia de fuego, lo que nos permitió observar su comportamiento en contextos variados.

Podemos comprobar en la Figura 5.15 cómo el modelo es capaz de detectar patrones de fuego similares a los presentes en el conjunto de entrenamiento, como en las fogatas de los ejemplos 1 y 3 o en el incendio en un barco en el ejemplo 7. Sin embargo, en otras imágenes que no contienen incendios, especialmente aquellas capturadas en el entorno de PLOCAN, el modelo tiende a marcar regiones como fuego, produciendo falsos positivos que podrían representar un riesgo en un sistema de monitoreo.

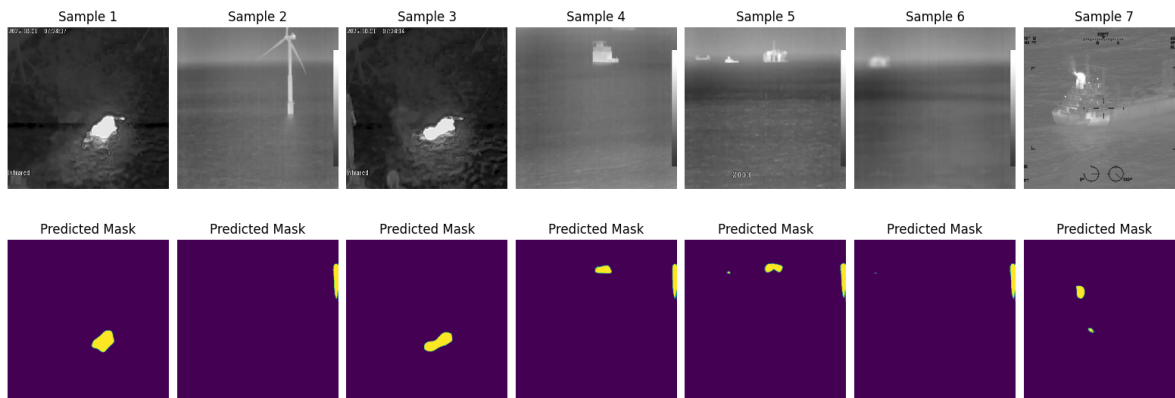


Ilustración 5.15: Resultados de las máscaras binarias predichas por el modelo de segmentación U-Net en imágenes de incendios y otros entornos que no presentan fuego.

Estos resultados indican que el modelo U-Net se beneficiaría de un conjunto de datos IR más extenso y diversificado, así como de un refinamiento adicional en los parámetros para mejorar la precisión en escenarios infrarrojos. Si bien el modelo muestra una capacidad básica para identificar áreas de fuego, la escasez de datos y las inconsistencias observadas sugieren que esta es solo una aproximación inicial. Esto nos lleva a la conclusión de que era necesario emprender una nueva fase de recolección de datos para fortalecer el rendimiento del modelo en situaciones reales.

5.3. Creación de Imágenes Sintéticas usando Redes Generativas Adversariales (GANs)

Durante el desarrollo de este proyecto, nos enfrentamos a un problema crítico que ralentizó el progreso: la dificultad de obtener imágenes térmicas reales, especialmente de incendios en la escala térmica **WhiteHot**. Este problema fue particularmente relevante dado que muchas investigaciones en el área de detección de incendios se centran en el uso de sensores ópticos, como cámaras RGB, que proporcionan un acceso más amplio a conjuntos de datos. Sin embargo, muy pocos estudios ofrecen conjuntos de datos que incluyan simulaciones de incendios en infrarrojo (IR), y mucho menos en escenarios específicos como entornos portuarios o marítimos.

Si bien nuestro objetivo es desarrollar un modelo general que sea aplicable a distintos entornos, el acceso que tenemos a estas áreas específicas hace que sea especialmente útil para nosotros contar con imágenes térmicas de incendios en estos contextos. Crear simulaciones reales de incendios no solo sería una tarea compleja y costosa, sino que también requeriría condiciones controladas y seguras, lo cual implica costos humanos, materiales y logísticos considerables. Debido a estos desafíos, gran parte de nuestro esfuerzo se enfocó en recolectar datos térmicos y explorar métodos alternativos para generar imágenes sintéticas que pudieran ser utilizadas en las pruebas.

Una de las soluciones más prometedoras que exploramos fue el uso de **Redes Generativas Adversariales (GANs)**, una técnica poderosa para la generación de datos sintéticos. Las GANs son especialmente útiles cuando existe una carencia de datos reales, ya que permiten crear imágenes nuevas y realistas a partir de los patrones aprendidos de un conjunto de datos limitado. Esto es particularmente relevante en nuestro caso, donde la falta de imágenes de incendios en IR, y más específicamente en la escala **WhiteHot**, representa una barrera significativa para el desarrollo de modelos robustos.

Las GANs constan de dos redes neuronales que compiten entre sí: el **generador**, que intenta crear imágenes sintéticas lo más parecidas posible a las reales, y el **discriminador**, que evalúa si una imagen es real o generada. A través de este proceso de competencia, ambas redes mejoran con el tiempo, y el generador es capaz de producir imágenes cada vez más realistas. Con esta técnica en mente, probamos varias implementaciones para evaluar su rendimiento y resultados, con el fin de determinar si las imágenes sintéticas generadas eran adecuadas para su uso en el entrenamiento y validación de nuestros modelos.

5.3.1. Revisión de Datasets de Imágenes Térmicas en el Contexto de Incendios y Entornos Marítimos

Para poder poner a prueba y evaluar correctamente los modelos desarrollados en este proyecto, fue crucial obtener más datos en contextos térmicos, específicamente en entornos marítimos. Debido a la escasez de conjuntos de datos específicos para esta tarea, nos enfocamos en investigar diferentes fuentes como artículos académicos, repositorios especializados y plataformas como **Kaggle**³ y **Roboflow**⁴. El objetivo fue recopilar imágenes térmicas que pudieran proporcionar un mayor respaldo en las fases de entrenamiento y validación de los modelos, especialmente en lo que respecta a la detección de anomalías térmicas en escenarios variados.

Conjunto de datos obtenidos manualmente de QAISC y PLOCAN Este conjunto de datos fue recopilado manualmente en el entorno del centro comercial El Mirador, cerca de PLOCAN, con imágenes térmicas obtenidas en diferentes momentos del día y de diversos objetos, como barcos, plataformas y un aerogenerador. Estas imágenes proporcionan una base variada en cuanto a condiciones lumínicas y objetos en entornos marítimos, lo que resulta útil para entrenar y validar modelos en un contexto específico.

FLAME Dataset El conjunto de datos FLAME, como ya se mencionó anteriormente, está centrado en incendios en zonas boscosas y fue capturado mediante cámaras térmicas montadas en drones. Contiene imágenes en infrarrojo (IR) que proporcionan una vista aérea de incendios controlados en entornos naturales, siendo particularmente útil para entrenar modelos que detectan patrones de fuego en vegetación.

³Disponible en: <https://www.kaggle.com/>

⁴Disponible en: <https://universe.roboflow.com/>

MassMIND: Massachusetts Maritime INfrared Dataset El dataset MassMIND fue desarrollado para aplicaciones en el dominio marítimo. Consiste en 2916 imágenes capturadas en el puerto de Boston y áreas aledañas durante un periodo de dos años. Las imágenes están en infrarrojo de onda larga (LWIR) y cubren una gran diversidad de escenas, incluyendo aguas turbulentas, neblina, cielos despejados y zonas costeras con tráfico marítimo. Este conjunto de datos se segmenta a nivel de píxeles y está anotado en clases como agua, cielo, obstáculos, puentes y entidades vivientes, lo que lo hace ideal para entrenar modelos en ambientes marítimos complejos [35].

HIT-UAV: A High-altitude Infrared Thermal Dataset El HIT-UAV es un conjunto de datos de 2898 imágenes térmicas infrarrojas capturadas desde vehículos aéreos no tripulados (UAVs) a gran altitud. Este dataset incluye una amplia variedad de escenas, lo que lo convierte en una herramienta útil para entrenar modelos de detección y seguimiento de objetos en escenarios aéreos a gran escala. Las imágenes están tomadas desde diferentes alturas y ofrecen perspectivas térmicas de varios tipos de terrenos, con énfasis en el análisis térmico de grandes áreas. Obtuvimos este conjunto de datos desde la plataforma Kaggle, lo que nos permitió disponer de imágenes térmicas reales en otros contextos [48].

Marine Image Thermal Dataset Este conjunto de datos se obtuvo de Roboflow y contiene 131 imágenes térmicas capturadas en entornos portuarios. Es un conjunto más reducido, pero proporciona imágenes reales en situaciones como plataformas marítimas y barcos, lo que complementa los otros datasets al ofrecer datos de un contexto especializado que podría ser relevante para nuestro proyecto [6].

5.3.2. Aplicación de CycleGAN para la Traducción de Imágenes

Uno de los métodos que exploramos para la transformación de imágenes fue **CycleGAN**. Esta técnica se basa en redes generativas adversariales (GANs) y fue desarrollada por Jun-Yan Zhu et al. [55], presentándose como una solución innovadora para la traducción de imágenes no emparejadas. CycleGAN es particularmente útil en situaciones donde no se cuenta con pares de imágenes equivalentes en diferentes dominios, como en nuestro caso, donde queríamos traducir imágenes RGB que presentaran incendios a imágenes térmicas.

El código de CycleGAN ⁵ es de acceso libre y puede encontrarse en el repositorio oficial de GitHub proporcionado por los autores. Este código fue una base sólida para nuestras pruebas y adaptaciones, permitiéndonos entrenar un modelo que intentaba traducir imágenes térmicas en escala WhiteHot a su equivalente RGB, y viceversa. Utilizando el conjunto de datos recopilados previamente junto a otros datasets, pudimos entrenar y probar la implementación del modelo para intentar generar imágenes acordes a cómo podría verse en una cámara térmica real.

⁵Disponible en: <https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix>

Pruebas y resultados de las imágenes de CycleGAN



Ilustración 5.16: Resultados del primer intento de transformación de dominios RGB a imágenes térmicas mediante CycleGAN, siendo la fila superior las imágenes RGB reales y la fila inferior la transformación generada.

Realizamos una primera prueba para evaluar el rendimiento del modelo entrenado, utilizando las imágenes RGB como **dominio A** y las imágenes térmicas como **dominio B**. El entrenamiento se llevó a cabo durante **200 épocas**, y algunos de los resultados obtenidos se muestran en la Figura 5.16.

A primera vista, observamos que, aunque las zonas de colores cálidos como el fuego se transforman en tonos más claros, acercándose al blanco, en ningún momento se asemejan a una imagen térmica real. En lugar de ello, las imágenes generadas parecen una conversión a escala de grises, donde los colores menos intensos se vuelven oscuros (negros o grises) y los más intensos se transforman en blancos. Aunque esto imita en parte la escala *WhiteHot*, no funciona de manera precisa, ya que una imagen térmica se basa en la temperatura de los objetos. Como no teníamos acceso a los umbrales de temperatura para guiar la transformación, CycleGAN interpretó erróneamente algunos objetos, como los barcos, convirtiéndolos en grises en lugar de blancos.

Otro factor que contribuye al ruido en las imágenes generadas es la forma en que CycleGAN aprende de los patrones en las imágenes, basándose principalmente en el color, formas o contenido para pasar de un dominio a otro. Al no tener en cuenta la temperatura, el modelo puede aprender características irrelevantes de las imágenes, lo que añade distorsión en las transformaciones. Aunque una posible mejora sería aumentar el número de imágenes utilizadas para entrenar el modelo, el caso específico de transformar imágenes RGB a IR es particularmente delicado, ya que involucra información de temperatura que no está presente en las imágenes RGB.

A pesar de estos problemas, dedicamos tiempo a optimizar el modelo, ajustando parámetros como el número de épocas, modificando hiperparámetros, agregando más imágenes al conjunto de datos, e incluso aumentando el tamaño de las imágenes generadas de 256×256 a 1024×1024 . Sin embargo, ninguna de estas modificaciones resultó en mejoras significativas.

Dado el tiempo y los recursos invertidos, y al no obtener los resultados esperados, decidimos explorar otros métodos para simular correctamente las anomalías térmicas en imágenes IR.

5.3.3. Aplicación de Pix2Pix para la Traducción de Imágenes

Otro enfoque que exploramos para la traducción de imágenes fue **Pix2Pix**, un modelo basado en **Redes Generativas Adversariales Condicionales** (cGANs), desarrollado por Phillip Isola et al. [20]. A diferencia de CycleGAN, que es útil cuando no hay pares de imágenes entre los dominios, Pix2Pix se enfoca en situaciones donde se tienen pares de imágenes de entrada y salida que corresponden entre sí. Esto lo convierte en una herramienta valiosa para tareas como la conversión de imágenes RGB a térmicas, cuando contamos con ejemplos en ambos dominios.

El código de Pix2Pix ⁶ y ⁷ está disponible en GitHub y es ampliamente utilizado para aplicaciones de traducción de imágenes, incluyendo la síntesis de imágenes a partir de mapas de etiquetas, la generación de fotografías a partir de mapas de contornos, y otras tareas. En nuestro caso, probamos este modelo con el objetivo de transformar imágenes RGB de incendios en sus equivalentes térmicos.

El funcionamiento de Pix2Pix se basa en una red generadora que crea imágenes sintéticas y una red discriminadora que aprende a diferenciar entre imágenes reales y generadas. A diferencia de los GANs tradicionales, Pix2Pix agrega una **pérdida L1** para mejorar la calidad de las imágenes generadas, lo que permite reducir el desenfoque y obtener imágenes más nítidas, ajustándose mejor a las características deseadas del dominio de salida.

Pruebas y resultados de las imágenes de Pix2Pix

Como mencionamos anteriormente, la principal diferencia de Pix2Pix es que el conjunto de datos utilizado para entrenar el modelo debe estar estrictamente emparejado. Es decir, en nuestro caso, es necesario contar con imágenes RGB y sus correspondientes imágenes térmicas tomadas simultáneamente. Para generar este conjunto de datos, capturamos vídeo de manera simultánea tanto con el sensor óptico como con el sensor térmico, utilizando la cámara PTZ del entorno de PLOCAN. Posteriormente, extraímos los frames correspondientes y los emparejamos, alineando las imágenes RGB con sus respectivas versiones térmicas, lo que nos permitió crear un dataset para el entrenamiento del modelo, como los ejemplos de la Figura 5.17.

⁶Disponible en: <https://github.com/phillipi/pix2pix>

⁷Disponible en: <https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix>



Ilustración 5.17: Imágenes RGB asociadas a su respectivo frame en IR obtenidas mediante la cámara PTZ proporcionada por QAISC y PLOCAN. Las diferencias mínimas entre imágenes del mismo frame se deben a variaciones en las lentes óptica y térmica, como el tamaño del sensor o la distorsión.

Una vez emparejadas las imágenes, preparamos un conjunto de datos compuesto por un total de **468 pares** de imágenes para el entrenamiento. Aunque el número de imágenes no era muy elevado, este dataset nos permitió obtener una primera visión de los resultados y evaluar la eficacia de esta aproximación.



Ilustración 5.18: Resultados de transformación de dominios RGB a imágenes térmicas mediante Pix2Pix, siendo la fila superior las imágenes RGB reales y la fila inferior la transformación generada.

Tras entrenar el modelo durante 200 épocas, observamos los resultados obtenidos, similares a los que se muestran en la Figura 5.18. Aunque en algunas imágenes parece que el modelo intenta generar formas, los resultados fueron mayormente pobres, con una cantidad considerable de ruido que dificulta la interpretación de las imágenes. Las formas resultantes no se asemejan a representaciones reales, lo que sugiere que el modelo no logró capturar adecuadamente las características del dominio térmico.

A pesar de realizar varias pruebas adicionales, los resultados no mostraron mejoras significativas. Es posible que, con un mayor número de datos emparejados y un ajuste más preciso de los hiperparámetros y el número de épocas, se podrían obtener resultados más satisfactorios. Sin embargo, dado el tiempo y los recursos que esto implicaría, decidimos explorar otras alternativas para alcanzar nuestro objetivo.

5.3.4. Aplicación de MUNIT para la Traducción de Imágenes

El último enfoque que exploramos para la traducción de imágenes mediante GAN fue el modelo **MUNIT (Multimodal Unsupervised Image-to-Image Translation)**, propuesto por Xun Huang et al. [18]. Este modelo se basa en una arquitectura GAN que permite la traducción entre dominios sin la necesidad de pares de imágenes emparejadas estrictamente. Además, tiene la capacidad de generar múltiples resultados posibles en el dominio de salida mediante la separación de la representación de contenido y estilo en las imágenes.

A diferencia de otros métodos como Pix2Pix, que requieren conjuntos de imágenes emparejadas, MUNIT opera en un entorno no supervisado, lo que lo convierte en una solución ideal para tareas donde es difícil obtener imágenes emparejadas, como la conversión de imágenes RGB a infrarrojo (IR). El modelo utiliza una codificación separada para el contenido y el estilo de las imágenes, permitiendo generar diferentes versiones de una imagen en el dominio de destino simplemente ajustando el código de estilo.

El código de MUNIT utilizado en este proyecto ⁸ se basa en una implementación modificada por Dong-Guw Lee, Myung-Hwan Jeon, Younggun Cho y Ayoung Kim para la traducción de imágenes de RGB a IR, publicada en 2023 [28]. Este modelo preserva los detalles clave de las imágenes de entrada, utilizando una técnica de traducción guiada por bordes para asegurar la coherencia en los detalles dinámicos de las imágenes transformadas. Viendo los resultados prometedores obtenidos por este equipo, decidimos probar su implementación en nuestro contexto para generar imágenes térmicas más precisas a partir de imágenes RGB.

Pruebas y resultados de las imágenes de MUNIT

Para las pruebas con el modelo MUNIT, utilizamos los pesos preentrenados que nos proporcionaron los autores. Estos pesos fueron entrenados con una configuración específica que incluía 60,000 iteraciones, un tamaño de lote de 1, y un decaimiento de peso de 0.001. El modelo fue optimizado utilizando el algoritmo Adam, con parámetros de $B1 = 0.5$ y $B2$

⁸Disponible en: <https://github.com/RPM-Robotics-Lab/sRGB-TIR>

= 0.999, y una tasa de aprendizaje inicial de 0.0001. Además, se implementó una política de decaimiento en la tasa de aprendizaje, con un factor de gamma de 0.5. Las imágenes de entrada, tanto las sintéticas en RGB como las térmicas, tenían un tamaño de 640 x 400.

Con esta configuración, realizamos las pruebas utilizando el modelo preentrenado para llevar a cabo la inferencia con varias imágenes. Como hemos mencionado anteriormente, una de las características distintivas de MUNIT es su capacidad para generar múltiples salidas a partir de una sola imagen de entrada, gracias a la separación entre el contenido y el estilo. Durante las pruebas, evaluamos esta característica para ver cómo varían las posibles salidas, observando los distintos resultados generados para una misma imagen de entrada.

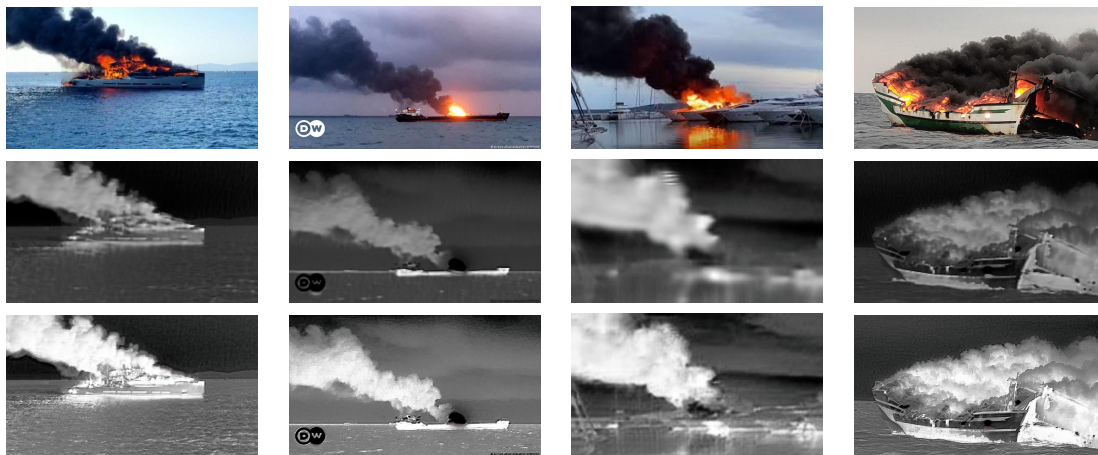


Ilustración 5.19: Resultados de transformación de dominios RGB a imágenes térmicas mediante MUNIT, siendo la primera fila las imágenes originales RGB y cada una de las otras filas un resultado distinto generado en una misma inferencia.

Como se puede observar en la Figura 5.19, los resultados obtenidos a simple vista parecen satisfactorios, ya que varios elementos se transforman en tonos blancos, grises y oscuros, lo que en principio podría interpretarse como una conversión térmica adecuada. Sin embargo, un análisis más detallado revela que el modelo no cumple con el objetivo principal de convertir imágenes de RGB a infrarrojo. En lugar de realizar una transformación térmica precisa, lo que parece hacer es invertir los colores claros y oscuros.

Este comportamiento es particularmente evidente en las imágenes donde aparece fuego brillante en la versión original en RGB. Al ser transformadas, estas zonas que deberían aparecer blancas en una imagen térmica real, se vuelven negras. Del mismo modo, áreas que originalmente eran oscuras, como el humo o los barcos, pasan a ser blancas, lo que indica que la red está aplicando una especie de inversión de color, más cercana a una conversión en escala de grises que a una verdadera traducción de RGB a IR.

Además, intentamos entrenar el modelo con nuestro propio conjunto de imágenes RGB e IR, pero los resultados no mejoraron. Para confirmar nuestra hipótesis, revisamos el apartado de “**Issues**” del repositorio del proyecto, donde encontramos un hilo titulado “These converted images are not thermal images”. En este hilo, otro usuario mencionaba el mismo

problema: las imágenes convertidas no eran térmicas, sino que se trataba de una inversión de colores. A partir de este punto, decidimos explorar otras alternativas para simular condiciones adversas en la escala *WhiteHot*, ya que los múltiples intentos con las GANs no lograron producir los resultados deseados.

5.4. Generación de Imágenes Sintéticas mediante Modelos de Texto a Imagen

Tras probar diversos métodos de generación de imágenes, como las GANs, y no obtener los resultados satisfactorios necesarios para simular de forma precisa imágenes en infrarrojo (IR) con anomalías térmicas, continuamos investigando otras alternativas actuales que pudieran ayudarnos a alcanzar nuestro objetivo. En este punto, la creación de imágenes IR que representaran de forma realista fenómenos como fuego y humo seguía siendo un desafío considerable.

En los últimos años, ha habido grandes avances en modelos de procesamiento de lenguaje natural y generación de contenido a partir de texto, tanto para generar respuestas textuales como para producir imágenes. Modelos como los **Large Language Models (LLMs)**, que incluyen a **GPT** y otros, han revolucionado la forma en que interactuamos con la tecnología, proporcionando generación de texto y, en algunos casos, otros tipos de contenido, como imágenes. Esto llevó a un aumento significativo en el desarrollo de modelos especializados en la generación de imágenes a partir de texto, ofreciendo nuevas posibilidades para nuestro trabajo.

Inicialmente, decidimos explorar algunos de los modelos más conocidos, como **DALLE**, que ya había demostrado ser efectivo en la creación de imágenes realistas. Sin embargo, nuestras primeras pruebas, realizadas en entornos web que utilizaban este modelo, no produjeron resultados satisfactorios. A pesar de especificar con claridad que necesitábamos imágenes en escala térmica, los resultados no lograban representar adecuadamente las características propias de las imágenes IR directamente con el texto introducido. Por esta razón, priorizamos el uso de GANs en las fases iniciales del proyecto.

No obstante, conscientes de que los modelos de generación de imágenes por texto seguían avanzando, decidimos seguir explorando estas herramientas. Durante esta fase, encontramos una plataforma web llamada **Getimg.ai**⁹. Esta plataforma nos llamó la atención por su amplio conjunto de herramientas para la generación de imágenes y vídeos a partir de texto, así como por su facilidad de uso. Además, contaba con una gran base de usuarios, generación por tokens mediante API, una documentación detallada y bien estructurada, y ejemplos en vídeo que facilitaban la comprensión de sus funcionalidades.

Lo más interesante para nuestro proyecto fue que Getimg.ai ofrecía la posibilidad de generar hasta **1000 imágenes** de manera gratuita, lo que nos permitió realizar pruebas exhaustivas antes de comprometernos con una suscripción. Este entorno parecía ideal para

⁹Disponible en: <https://getimg.ai/>

continuar experimentando y evaluar si podía brindarnos mejores resultados en la simulación de anomalías térmicas que otros métodos no habían logrado.

5.4.1. Interfaz y Funcionalidades de Texto a Imagen

La herramienta principal que utilizamos en la plataforma fue el “AI Generator”, que permite la generación de imágenes mediante modelos entrenados para esta tarea, basándose en la entrada de texto o *prompts*. Además, es posible modificar ciertos parámetros para manipular el resultado de las imágenes generadas según las necesidades del proyecto.

Inicialmente, trabajamos con la interfaz “Essential”, que nos permitía elegir el modelo de generación. En este caso, probamos con el modelo **FLUX.1 [schnell]**, un modelo relativamente nuevo que mostraba un rendimiento prometedor en la generación de imágenes¹⁰. Además de poder escribir el prompt, también podíamos seleccionar el *aspect ratio* de las imágenes.

Usamos el modelo **FLUX** y como prompt empleamos el siguiente texto:

“Generate a hyperrealistic image of a burning ship or port area, captured through a thermal camera using the Whitehot scale. In this scale, the hottest areas, such as the burning ship and the intense flames, should appear in bright white. The smoke should be depicted in various shades of gray, representing cooler temperatures compared to the flames. The surrounding water, horizon, and other cooler elements should be in darker shades, ranging from dark gray to black. The image should convey a stark contrast between the white-hot flames, the gray smoke, and the dark, cooler environment, highlighting the intensity of the fire.”

Obteniendo resultados como los mostrados en la Figura 5.20, que, como podemos observar, no son los esperados ni reflejan adecuadamente lo que buscábamos en una imagen térmica.



Ilustración 5.20: Resultados obtenidos con el modelo FLUX.1 [schnell] utilizando el prompt anterior para generar imágenes IR sintéticas de incendios en barcos.

¹⁰Disponible en: <https://github.com/black-forest-labs/flux>

Como los resultados iniciales no fueron los esperados, decidimos probar la interfaz avanzada y utilizar un modelo especializado en la generación de imágenes hiperrealistas llamado **JuggernautXL X**. Al aplicar el mismo prompt, configuramos el proceso con 30 pasos de generación, una *guidance scale* de 7 y el sampler *Euler*. Aunque las imágenes obtenidas fueron visualmente más detalladas y de mejor calidad, seguían sin cumplir completamente con nuestro objetivo, ya que aún se generaban en escala de grises en lugar de reproducir la escala *WhiteHot* propia de las imágenes térmicas.

En cuanto a los parámetros utilizados, los **30 pasos** de generación que configuramos se refieren a las iteraciones que el modelo realiza para refinar la imagen. Cuanto mayor es el número de pasos, más oportunidades tiene el modelo de mejorar la calidad de los detalles y la coherencia en la imagen generada. Sin embargo, esto también aumenta el tiempo de procesamiento, lo cual debe balancearse según las necesidades del proyecto.

Por otro lado, la **guidance scale** es un parámetro clave que define hasta qué punto el modelo sigue fielmente la descripción proporcionada en el prompt. En nuestro caso, al fijar este valor en 7, buscamos forzar al modelo a ajustarse lo máximo posible a las indicaciones específicas del texto. Aunque esto mejora la fidelidad al prompt, también puede limitar la flexibilidad del modelo para generar imágenes más creativas o variadas, lo que podría ser necesario en otros contextos.

Finalmente, el **sampler Euler** es el algoritmo que seleccionamos para guiar el proceso de generación de píxeles en la imagen final. Existen varios tipos de samplers, y el Euler es conocido por su equilibrio entre precisión y la calidad visual de las imágenes generadas, proporcionando buenos resultados en la mayoría de las aplicaciones de generación de imágenes a partir de texto.

A pesar de estos ajustes y de la mejora visual obtenida, las imágenes seguían sin reflejar adecuadamente las características térmicas que buscábamos, confirmando que esta aproximación no era adecuada para nuestro caso (Figura 5.21).



Ilustración 5.21: Resultados obtenidos con el modelo Juggernaut XL X utilizando el prompt anterior para generar imágenes IR sintéticas de incendios en barcos.

Ante los resultados obtenidos, decidimos consultar las guías ofrecidas por la plataforma para explorar en mayor profundidad la opción de **“Image references”**. Esta funcionalidad permite utilizar una o varias imágenes de referencia y seleccionar un tipo específico de transformación basada en técnicas como **“IP Adapter”** y **“ControlNet”**. Estas opciones incluían ejemplos de generación de imágenes a partir de otras similares (como en **“Image to Image”**), basándose en el contenido visual (**“Content”**) o utilizando contornos y formas con diferentes niveles de detalle, como en las opciones de **“Hard”** y **“Soft Edges”**, entre otras. Sin embargo, uno de los modos que consideramos más útil fue el de **“Style”**. Según aprendimos en las guías, esta opción permite extraer valores de color, brillo y estilos artísticos de la imagen de referencia, lo que en nuestro caso, al simular una escala térmica como la *WhiteHot*, podría ser de gran utilidad.

Decidimos entonces probar el mismo prompt y parámetros que habíamos utilizado anteriormente, pero esta vez añadiendo una imagen de referencia obtenida de la cámara PTZ en el entorno de PLOCAN, configurando el valor de similitud de estilo en **135**. Finalmente, logramos obtener una imagen que podía considerarse como una simulación sintética bastante precisa de una imagen térmica en la escala que estábamos buscando (Figura 5.22). Aunque esta imagen aún presentaba algunas diferencias con respecto a una imagen térmica real, como variaciones en el brillo u otras características a las que no pudimos acceder debido a las limitaciones del modelo y del material original, logramos alcanzar un punto en el que las imágenes generadas se acercaban mucho más a nuestros objetivos iniciales.

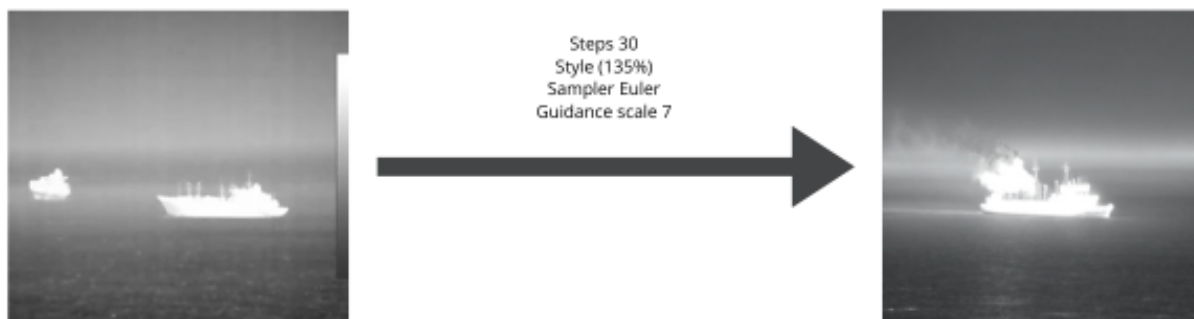


Ilustración 5.22: Imagen que simula la escala *WhiteHot* mediante el modelo de Juggernaut XL X en base al estilo de una imagen de referencia.

A partir de este punto, comenzamos a ajustar los parámetros del modelo, variando el número de pasos, utilizando nuevas imágenes de referencia y explorando diferentes tipos de transformaciones para generar imágenes más variadas. También experimentamos con una amplia gama de prompts, algunos más simples y otros más elaborados, para obtener resultados más específicos. Entre los prompts utilizados, destacamos descripciones como:

- *“A big cargo ship burning with white fire flames and light gray smoke in the middle of the sea.”*

- *“A small ship burning with fire and smoke in the middle of the sea in whitehot scale. Light gray smoke.”*
- *“A wind turbine burning with fire and smoke in middle of the sea in whitehot scale.”*
- *“Coastal thermal image of a forest fire. The fire line is a white band against the darker landscape. Smoke appears as lighter gray plumes rising into the black sky. The sea is a uniform dark mass, contrasting with the white shoreline fire.”*
- *“Thermal image of a fire at an oil refinery. Multiple white areas indicate fires among the complex structures. Smoke appears as lighter gray plumes rising upward. Unaffected parts of the refinery show in various shades of gray. Emergency vehicles are visible as white shapes moving around the perimeter.”*

Estos prompts nos permitieron generar imágenes con un mayor nivel de detalle (Figura 5.23), acercándonos más a nuestras expectativas. Gracias a esta combinación de pruebas y ajustes, logramos generar aproximadamente **1430** imágenes, de las cuales seleccionamos **855** para continuar con nuestras evaluaciones.



Ilustración 5.23: Ejemplo de imágenes finales generadas por Juggernaut XL X a partir de algunos de los anteriores prompts.

5.5. YOLO para la detección de anomalías térmicas

Una vez que logramos crear imágenes sintéticas, el siguiente paso fue evaluarlas para determinar si podían ser útiles en nuestro objetivo de detectar situaciones anómalas relacionadas con incidentes térmicos. Para esto, decidimos cambiar de enfoque y optar por otro modelo de detección. Aunque métodos de segmentación como U-Net habrían sido útiles en esta tarea, queríamos explorar diferentes soluciones y observar cómo se comportaban con nuestro conjunto de datos.

Finalmente, optamos por utilizar el modelo **YOLO** (*You Only Look Once*), uno de los más populares para la detección en tiempo real debido a su capacidad para realizar detecciones rápidas y precisas en una sola pasada por la imagen. A diferencia de otros métodos de detección, que dividen la imagen en regiones múltiples y analizan cada una por separado, YOLO considera la imagen completa en una única operación, mejorando significativamente la

velocidad sin comprometer la precisión. Su estructura está diseñada para identificar objetos en tiempo real, lo cual resulta ideal para aplicaciones que requieren rapidez, como la detección de anomalías térmicas en escenarios de emergencia.

Antes de poder entrenar y probar YOLO con nuestras imágenes sintéticas, preparamos cuidadosamente el conjunto de datos, ajustando las etiquetas y los formatos necesarios para asegurar una compatibilidad óptima con el modelo. Este paso era esencial para poder llevar a cabo el entrenamiento y las pruebas necesarias que verificarían si este método cumplía con nuestros objetivos y demostraba ser una solución prometedora para la detección de incidentes térmicos en entornos simulados.

5.5.1. Preparación y tratamiento del conjunto de imágenes

En las detecciones de YOLO, se utilizan *bounding boxes* para señalar la región de la detección. Este modelo emplea un formato específico de anotación que consiste en una fila con un número que indica la clase del objeto, seguido de las coordenadas del centro de la caja delimitadora (x, y) y, finalmente, el ancho y alto de la caja, todos ellos expresados en relación con el tamaño de la imagen.

<object-class>	<x-center>	<y-center>	<width>	<height>
1	0.245977	0.454995	0.491953	0.394893
0	0.468799	0.610420	0.094824	0.079941

Para crear estas anotaciones, utilizamos una herramienta llamada **CVAT** (*Computer Vision Annotation Tool*) ¹¹, una solución de código abierto que permite crear proyectos y tareas de anotación para diversos fines, como segmentación, detección y seguimiento de objetos. CVAT ofrece una interfaz intuitiva y fácil de manejar, lo que facilitó el proceso de creación de anotaciones en nuestro conjunto de datos (Figure 5.24).

En cuanto a las clases, decidimos trabajar con dos: **fuego** (clase 0) y **humo** (clase 1). Con esta estructura y las anotaciones preparadas, pudimos generar un conjunto de datos compatible con YOLO, adecuado para el entrenamiento y evaluación de detección de incidentes térmicos.

¹¹Disponible en: <https://www.cvat.ai/>

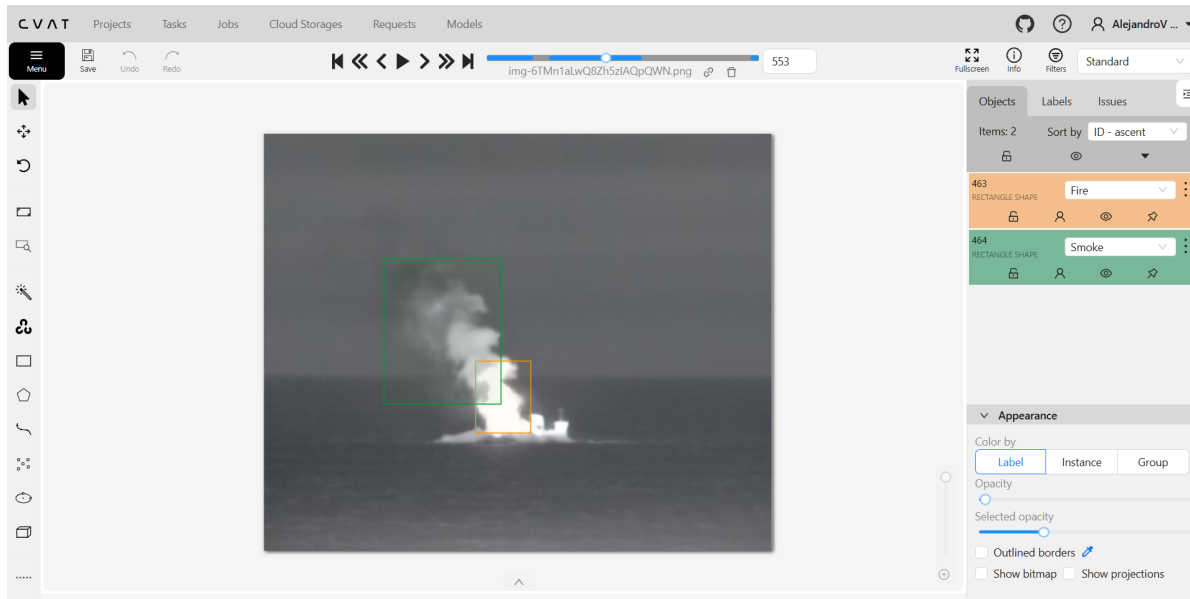


Ilustración 5.24: Interfaz web de CVAT para crear las cajas delimitadores de las clases de fuego (naranja) y humo (verde) del conjunto de datos.

Muestreo Negativo

Para preparar el conjunto de datos, también decidimos añadir imágenes que no presentaban incendios, pero que correspondían a entornos similares en infrarrojo (IR). Esta técnica se conoce como **negative sampling** o muestreo negativo. El negative sampling consiste en incluir ejemplos que no contienen el objeto o la anomalía que se desea detectar, lo cual es fundamental para entrenar modelos robustos. Al exponer al modelo tanto a ejemplos positivos (con incendios) como negativos (sin incendios), este aprende a diferenciar correctamente entre ambos casos y reduce la probabilidad de generar falsos positivos.

En nuestro caso, realizamos el muestreo negativo incluyendo tanto imágenes sintéticas como frames obtenidos del entorno de PLOCAN, asegurándonos de que estos ejemplos reflejaran escenarios reales sin la presencia de incendios, pero en condiciones ambientales y térmicas similares a las imágenes de entrenamiento que sí contenían incendios.

Eliminación de Imágenes Redundantes con dHash

Para evitar el posible sobreajuste en nuestro modelo, decidimos eliminar imágenes que fueran muy similares entre sí dentro del conjunto de datos. Esta reducción de redundancia asegura que el modelo no se “acostumbre” a patrones repetitivos, lo cual podría limitar su capacidad de generalización. Para identificar estas imágenes duplicadas o casi idénticas, empleamos la técnica de **dHash** (Differential Hashing), un método eficiente y rápido para detectar similitudes entre imágenes.

El **dHash** es una técnica que genera un valor hash (una representación numérica) para cada imagen, en función de las diferencias de brillo entre píxeles adyacentes. Este hash se calcula al convertir la imagen en escala de grises, redimensionarla a una pequeña matriz (por ejemplo, 9x8), y luego evaluar si el brillo de cada píxel es mayor o menor que su píxel vecino. Estas comparaciones producen una matriz de diferencias binarias, que se convierte en el valor hash final.

Cuando comparamos dos imágenes, sus valores hash permiten medir su grado de similitud mediante la *distancia de Hamming*, que es la cantidad de posiciones donde los hashes difieren. Si esta distancia es baja (por ejemplo, una similitud mayor al 85 %), las imágenes se consideran muy similares y una de ellas puede eliminarse.

Esta técnica de muestreo nos permitió identificar y remover imágenes redundantes, dejando solo aquellas que ofrecían variaciones significativas en el contenido, mejorando así la calidad y diversidad del conjunto de datos sin introducir ruido innecesario.

Con estas técnicas, una vez que consolidamos nuestro conjunto de datos personalizado, procedimos a dividirlo equitativamente en tres partes para los conjuntos de entrenamiento, validación y prueba, asignando el 70 %, 20 % y 10 % del total, respectivamente. Inicialmente, el conjunto completo constaba de 1324 imágenes

Conjunto de datos de incendios CORSICAN

Durante las fases finales del desarrollo de nuestro proyecto, logramos obtener acceso al **Conjunto de Datos de Incendios de Córcega** (*Corsican Fire Database*)¹², una base de datos gestionada por la Universidad de Córcega. Este conjunto de datos fue diseñado específicamente para apoyar investigaciones en la detección, monitoreo y análisis de incendios en entornos naturales y rurales, un tema de relevancia particular en la isla de Córcega debido a su vulnerabilidad a incendios forestales. La base de datos CORSICAN está orientada a mejorar las capacidades de los sistemas de detección de incendios y contribuir a la mitigación de desastres ambientales a través de tecnologías de visión artificial y sensores especializados.

La base de datos CORSICAN ofrece una amplia gama de información visual y espectral sobre incendios, y está estructurada en diferentes tipos de datos. En nuestro caso, utilizamos principalmente las imágenes en **Infrarrojo Cercano (NIR)** y sus correspondientes **máscaras de segmentación** que destacan las zonas afectadas por el fuego (Figura 5.25). Estas imágenes NIR son particularmente valiosas en la detección de incendios, ya que el infrarrojo cercano puede capturar firmas térmicas y detectar variaciones de temperatura en áreas específicas, permitiendo diferenciar entre vegetación y áreas en combustión.

¹²Disponible en: <https://cfdb.univ-corse.fr/index.php?menu=1>

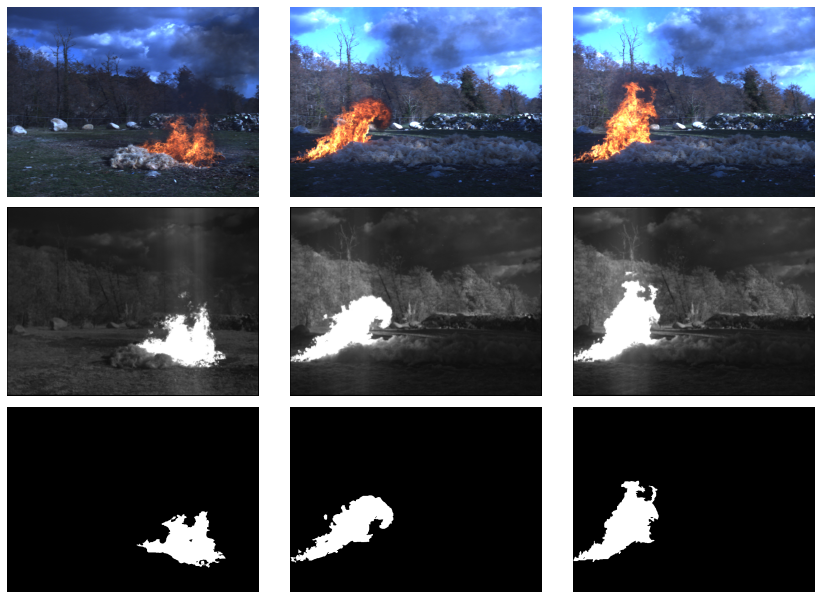


Ilustración 5.25: Ejemplos de imágenes de incendios del conjunto de datos CORSICAN. En la primera fila se presentan las imágenes en formato RGB, en la segunda fila se muestran sus correspondientes versiones en infrarrojo cercano (NIR), y en la tercera fila se encuentran las máscaras binarias que identifican las áreas afectadas por el fuego. Estas tres representaciones permiten una evaluación completa de las condiciones visuales y térmicas de los incendios.

5.5.2. Pruebas de Detección de Anomalías usando YOLOv8

Con los datos debidamente preparados, procedimos a implementar y evaluar el modelo **YOLOv8** [22], proporcionado por Ultralytics¹³. En esta etapa del proyecto, YOLOv8 destacaba como una de las versiones más avanzadas, con estadísticas de rendimiento superiores y un enfoque optimizado para la detección en tiempo real. Por estos motivos, decidimos emplear YOLOv8 en nuestras pruebas, confiando en que su arquitectura moderna y su rendimiento consolidado nos permitirían obtener resultados precisos y eficientes para la detección de anomalías térmicas en nuestras imágenes IR.

Inicialmente, comenzamos el proceso de entrenamiento probando varias versiones de **YOLOv8**, específicamente los modelos **YOLOv8n** (Nano), **YOLOv8s** (Small), **YOLOv8m** (Medium) y **YOLOv8l** (Large). Estos modelos varían en cuanto a complejidad y rendimiento: mientras que los modelos más ligeros (como YOLOv8n) son más rápidos y demandan menos recursos computacionales, los modelos de mayor tamaño, como YOLOv8l, ofrecen mayor precisión, aunque a costa de una mayor demanda de memoria y tiempo de procesamiento. Nuestra intención era evaluar estos modelos para identificar el que mejor equilibrara precisión y eficiencia en el contexto de detección de incendios y anomalías térmicas.

Los modelos YOLOv8 vienen preentrenados con el conjunto de datos **COCO** (Common Objects in Context) [29], que es uno de los datasets más utilizados en el campo de visión por

¹³Disponible en: <https://docs.ultralytics.com/es/models/yolov8/>

computador. COCO incluye alrededor de 330000 imágenes, de las cuales 200000 contienen anotaciones en 80 categorías diferentes, que abarcan desde animales y vehículos hasta objetos más específicos como sillas y relojes. Al estar entrenado con COCO, YOLOv8 adquiere un conocimiento amplio sobre la detección y clasificación de objetos variados, lo cual facilita su aplicación en otros contextos mediante **transfer learning** o aprendizaje por transferencia.

En este caso, entrenar un modelo preentrenado con COCO permite realizar un ajuste fino o **fine-tuning** sobre nuestro conjunto de datos específico. Mediante este proceso de *fine-tuning*, el modelo reutiliza las características generales aprendidas en COCO y se especializa en las nuevas clases del conjunto de datos de anomalías térmicas, lo que suele mejorar la precisión sin requerir un entrenamiento desde cero. Este enfoque no solo reduce el tiempo de entrenamiento, sino que también aprovecha la riqueza de conocimiento adquirida en un conjunto de datos grande y diverso, adaptándola al contexto específico de detección de fuego y humo en imágenes IR.

Con esta configuración inicial, procedimos a entrenar los distintos modelos de YOLOv8 con el objetivo de comparar su rendimiento en nuestro conjunto de datos. Utilizamos los parámetros predeterminados que proporciona YOLO, pero configuramos el entrenamiento para que se extendiera a **200 épocas** y establecimos un criterio de *patience* de **50 épocas**. Esto significa que, si el modelo no mejora después de 50 épocas consecutivas, el entrenamiento se detiene de forma anticipada para optimizar recursos y evitar un sobreajuste.

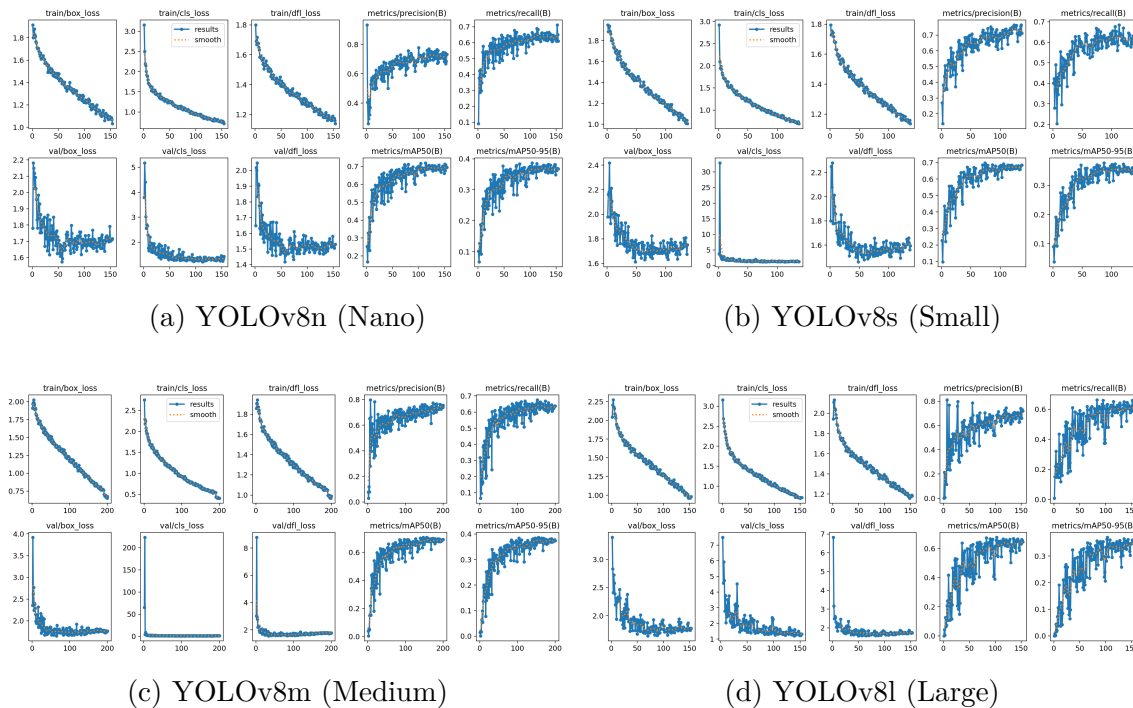


Ilustración 5.26: Comparación de los resultados de entrenamiento de los modelos YOLOv8: (a) YOLOv8n (Nano), (b) YOLOv8s (Small), (c) YOLOv8m (Medium), y (d) YOLOv8l (Large). Cada conjunto de gráficos muestra la evolución de las métricas clave, como la precisión y la pérdida, a lo largo de las épocas de entrenamiento para cada versión de YOLOv8, destacando sus fortalezas y limitaciones en función de su tamaño y capacidad computacional.

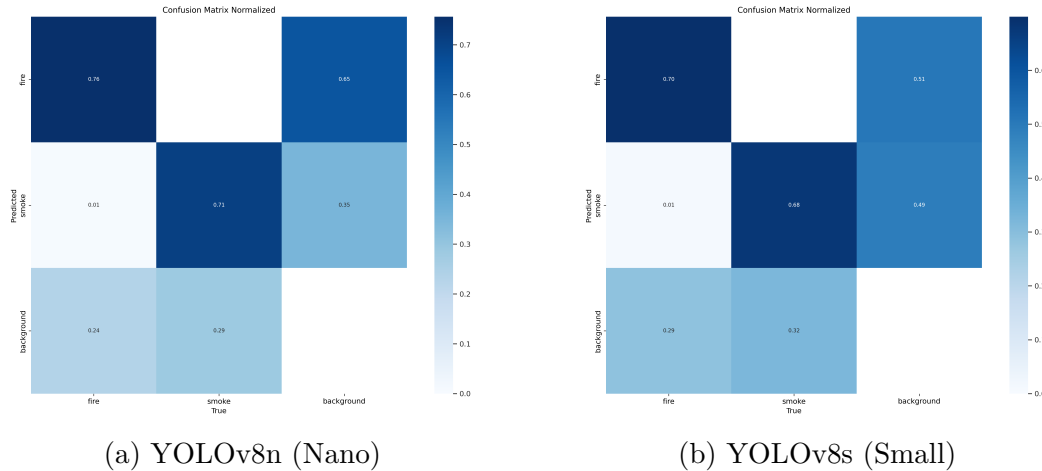


Ilustración 5.27: Matrices de Confusión Normalizadas de los Modelos YOLOv8n y YOLOv8s

Cuadro 5.3: Comparación del rendimiento total de todas las clases en la fase de evaluación de YOLOv8 para diferentes tamaños de modelo. Fuente: Datos obtenidos de los resúmenes de métricas.

Modelo	Precisión	Recall	mAP@50	mAP@50-95
YOLOv8-nano	72.60 %	66.32 %	71.53 %	38.73 %
YOLOv8-small	74.48 %	62.65 %	68.71 %	37.80 %
YOLOv8-medium	74.29 %	63.70 %	70.34 %	38.58 %
YOLOv8-large	68.94 %	63.22 %	67.74 %	37.07 %

Durante la fase de entrenamiento de los modelos YOLOv8 (Nano, Small, Medium y Large), se puede observar variaciones significativas en las métricas de rendimiento y pérdidas, estos presentados en la Figura 5.26.

En términos generales, todos los modelos mejoraron las métricas clave y redujeron las pérdidas a lo largo de las épocas de entrenamiento. El modelo Medium mostró la menor pérdida final, indicando una optimización efectiva. En cuanto a precisión, todos los modelos alcanzaron niveles superiores al 70 %. Sin embargo, al analizar el mAP@50 y el mAP@50-95, métricas clave para evaluar el rendimiento de detección durante el entrenamiento, el modelo Nano destacó sobre los demás. Estas métricas, *Mean Average Precision* (mAP) calculada a diferentes umbrales de Intersección sobre Unión (IoU), reflejan la precisión global del modelo: el mAP@50 considera un umbral de IoU del 50 %, mientras que el mAP@50-95 promedia valores de IoU entre 50 % y 95 %, lo que ofrece una evaluación más detallada y rigurosa del rendimiento. En estas métricas, el modelo Nano lideró, seguido en orden por el Medium, Small y, por último, el modelo Large.

En la fase de evaluación final, usando datos no vistos por los modelos, los resultados del rendimiento general se resumen en el Cuadro 5.3. Vemos que en general, todos los modelos mostraron un rendimiento adecuado en la detección de anomalías térmicas, destacando

especialmente el YOLOv8n (Nano). Aunque este modelo no presentó la mayor precisión en términos de identificación de objetos (siendo superado por el modelo Small, con un 74.48 %, y seguido de cerca por el Nano con un 72.60 %), sí destacó en las demás métricas. Por ejemplo, alcanzó un valor de Recall de 66.32 %, superando en 3.1 % al modelo Large.

Al analizar la matriz de confusión normalizada del modelo YOLOv8n (Nano) (Figura 5.27), se identifican patrones clave en la clasificación de las distintas clases de interés. Para la clase “fire”, el modelo logra una precisión del 76 % en la detección correcta de zonas con fuego, aunque presenta una confusión del 24 % con el fondo (“background”). Este nivel de precisión en la clase de fuego es adecuado, aunque la confusión con el fondo señala posibles áreas de mejora en la identificación de esta anomalía térmica específica.

En la clase “smoke”, el modelo alcanza una precisión del 71 %, con un 29 % de confusión hacia el fondo. Aunque la precisión en la detección de humo es ligeramente inferior a la de fuego, es consistente con la dificultad adicional que implica distinguir humo en un entorno complejo. Es importante resaltar que la confusión entre “fire” y “smoke” es mínima, con solo un 1 % de casos en los que el fuego se confunde con humo. Este bajo nivel de confusión entre las dos principales anomalías térmicas representa un buen indicador de que el modelo ha aprendido a diferenciar de manera efectiva estas características visuales, lo cual es esencial para la precisión general del sistema de detección.

Por otro lado, la clase “background” muestra una tasa de detección del 65 %, con falsos positivos distribuidos entre fuego (24 %) y humo (35 %). Esta dispersión en las detecciones del fondo sugiere que algunos patrones en las áreas de fondo comparten características con las anomalías térmicas, generando una confusión que podría reducirse en futuras mejoras del modelo.

A pesar de la expectativa de mejor rendimiento en los modelos más grandes (Medium y Large) debido a su mayor capacidad, estos no lograron superar de forma consistente a los modelos más pequeños en términos de precisión y eficiencia. Además, los modelos Medium y Large requieren más recursos computacionales y presentan tiempos de inferencia más prolongados, lo cual puede ser una desventaja en aplicaciones de tiempo real o en entornos con recursos limitados.

Finalmente, dada la combinación de precisión, rapidez y eficiencia de recursos, elegimos el modelo YOLOv8n (Nano) para las pruebas posteriores. Su balance óptimo entre rendimiento y consumo de recursos lo convierte en una elección favorable para nuestro objetivo de detectar anomalías térmicas en tiempo real.

5.5.3. Evaluación de YOLOv8n con datos sintéticos y reales

Con el modelo YOLOv8 Nano identificado como el de mejor rendimiento para nuestra tarea, decidimos realizar pruebas adicionales para evaluar si podíamos mejorar aún más sus resultados. Aprovechando el acceso reciente al conjunto de datos de incendios de CORSICAN, vimos una oportunidad de probar el modelo en un contexto de datos reales. Hasta este punto, habíamos logrado resultados satisfactorios entrenando el modelo con imágenes sintéticas

generadas mediante Juggernaut XL, enfocadas en simular incendios en una escala térmica. Sin embargo, el uso de datos reales de incendios en NIR (infrarrojo cercano) presentaba una posibilidad única de validar y comparar el rendimiento entre ambos tipos de datos.

El objetivo principal de esta prueba fue evaluar cuánto valor adicional aportaban los datos reales en comparación con los sintéticos y determinar si la combinación de ambos conjuntos podía mejorar la precisión de detección. Para ello, llevamos a cabo tres entrenamientos: uno exclusivamente con datos sintéticos previamente utilizados y ya entrenado, otro con sólo datos reales, y finalmente, un tercero con una combinación de datos reales y sintéticos.

Entrenamiento y evaluación de YOLOv8n con conjunto de datos completamente reales

En esta prueba, tomamos todas las imágenes NIR de fuegos (siendo esta la única clase que hay) del dataset de CORSICAN y las etiquetas correspondientes, siendo un total de **639** imágenes a las que añadimos otras de nuestro entorno de PLOCAN consiguiendo un dataset con un total de **963** imágenes repartidas en un 70 % para entrenamiento, 20 % para validación y el 10 % restante para test.

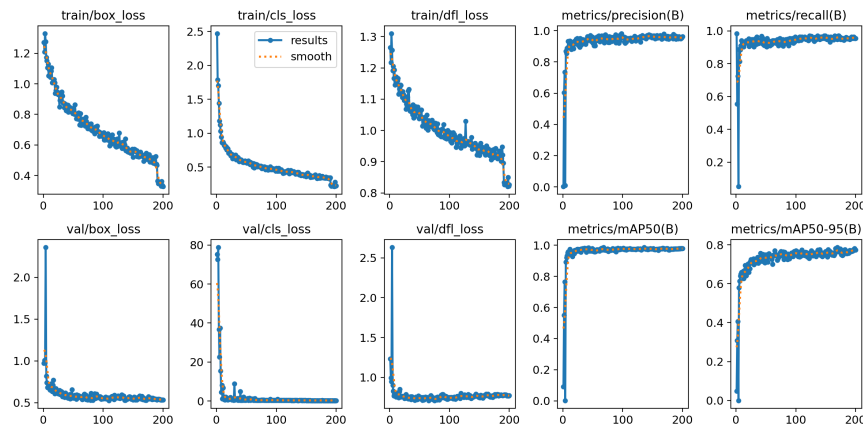


Ilustración 5.28: Rendimiento de las métricas de entrenamiento del modelo YOLOv8n con el conjunto de imágenes reales CORSICAN.

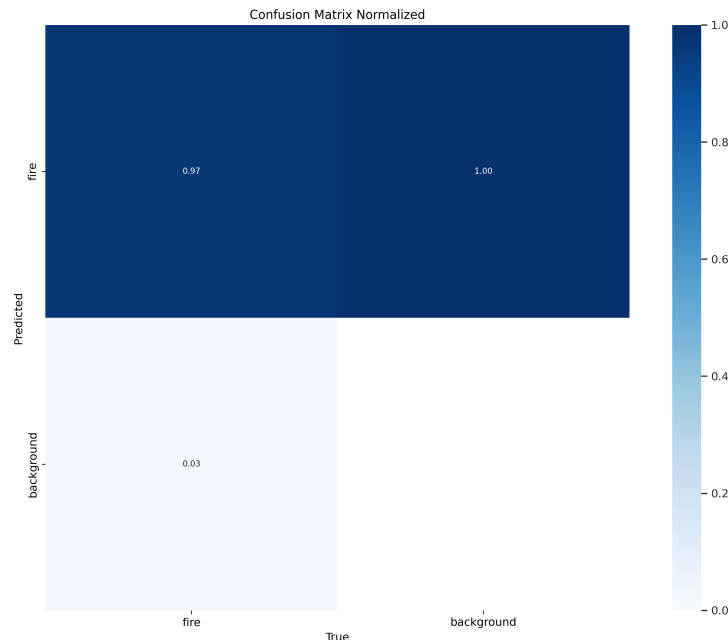


Ilustración 5.29: Matriz de confusión del set de test con las predicciones del modelo YOLOv8n en el conjunto de datos de imágenes reales de CORSICAN.

A simple vista, los resultados del modelo entrenado exclusivamente con imágenes reales de incendios parecen sobresalientes, tanto en el conjunto de entrenamiento como en el de prueba (Figura 5.28). Durante las épocas de entrenamiento, la pérdida disminuye progresivamente en ambas fases (entrenamiento y validación), mientras que las métricas de precisión, recall, mAP@50 y mAP@50-95 aumentan de forma significativa, alcanzando en muchos momentos valores cercanos al 100 %. Sin embargo, esta aparente perfección en los resultados es, en realidad, indicativa de un problema grave: el **overfitting** o sobreajuste.

Al observar el comportamiento de las métricas, se evidencia un cambio abrupto en las primeras épocas. Las métricas pasan de valores bajos a casi el máximo en un corto periodo, lo cual es una señal de que el modelo está memorizando el conjunto de entrenamiento sin generalizar bien a nuevos datos. La matriz de confusión del conjunto de prueba, mostrada en la Figura 5.29, confirma esto: el modelo alcanza una precisión del 97 %, lo cual podría interpretarse como un rendimiento casi perfecto. Sin embargo, esta precisión elevada no es necesariamente positiva, ya que apunta a que el modelo ha aprendido los patrones específicos del conjunto de entrenamiento (patrones visuales y características del entorno en imágenes reales de CORSICAN) en lugar de desarrollar una capacidad de generalización.

Este sobreajuste tiene sentido al considerar la composición del conjunto de datos: aunque se incluyeron algunas imágenes IR sin presencia de incendios, el resto de imágenes de incendios presentan características y patrones muy similares entre sí. Esto hace que el modelo identifique patrones visuales específicos de este conjunto, volviéndose altamente dependiente de ellos. Como resultado, el modelo es capaz de clasificar con alta precisión estas imágenes específicas, pero cuando se le presentan imágenes nuevas o de contextos distintos (como las imágenes sintéticas o las de otros escenarios de incendios en IR), falla en su capacidad de

predicción, como se observa en la Figura 5.30. Al intentar predecir con imágenes no incluidas en el entrenamiento, el modelo muestra un sesgo significativo debido a la falta de variabilidad en el conjunto de datos, limitando su capacidad para detectar incendios en contextos diferentes.



Ilustración 5.30: Ejemplos de detecciones predichas por el modelo YOLOv8n entrenado con sólo imágenes reales de CORSICAN. Se observa cómo patrones de fuego reales que son grabados de manera cercana pueden ser detectados pero imágenes de otros entornos no es capaz de distinguirlo.

Este análisis demuestra la importancia de contar con conjuntos de datos variados y representativos, ya que un modelo que muestra buenos resultados en un conjunto de datos homogéneo puede no ser útil en escenarios reales o en aplicaciones prácticas donde la diversidad de datos es crucial.

Entrenamiento y evaluación de YOLOv8n con conjunto de datos mezclados entre sintéticos y reales

Como prueba final de entrenamiento, evaluamos el rendimiento del modelo YOLOv8n en un conjunto de datos combinado que incluía tanto imágenes sintéticas de anomalías térmicas como imágenes reales extraídas del conjunto de datos CORSICAN. Este nuevo conjunto de datos constaba de un total de **1933** imágenes, distribuidas de manera estándar: un **70 %** para el entrenamiento, **20 %** para la validación y el **10 %** restante para el conjunto de prueba.

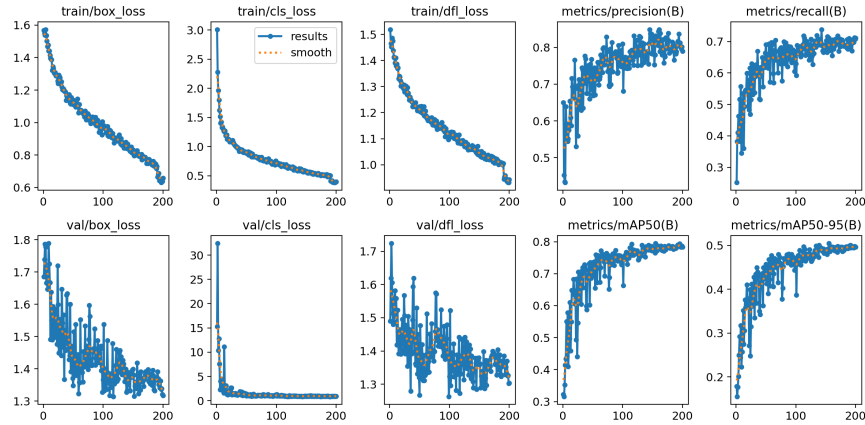


Ilustración 5.31: Rendimiento de las métricas de entrenamiento del modelo YOLOv8n con el conjunto de imágenes mezcladas.

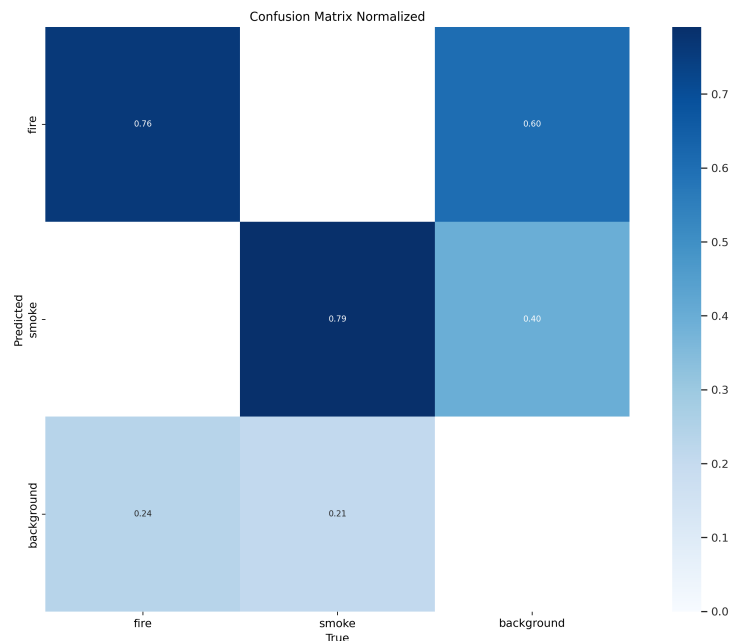


Ilustración 5.32: Matriz de confusión del set de test con las predicciones del modelo YOLOv8n en el conjunto de datos de imágenes mezcladas.

En esta prueba observamos mejoras significativas en el rendimiento del modelo. Al revisar los resultados obtenidos (ver Figura 5.31), se aprecia un excelente desempeño. A medida que avanzan las épocas, las pérdidas disminuyen progresivamente sin caídas abruptas, lo que indica un entrenamiento estable y controlado. Las métricas de precisión, *recall*, mAP50 y mAP50-95 también muestran un crecimiento gradual y sostenido, a diferencia del modelo entrenado únicamente con imágenes del conjunto CORSICAN, que presentaba picos de incremento repentinos, señal de sobreajuste. En este caso, el comportamiento de las métricas

es más consistente, con incrementos o ajustes en cada época, lo cual es ideal para una buena generalización.

En cuanto a los resultados, el modelo alcanza con frecuencia más del **80 %** tanto en precisión como en mAP50, mientras que la métrica mAP50-95 se sitúa en torno al **50 %** o incluso lo supera en algunas ocasiones, un logro que no se había conseguido en pruebas anteriores.

La matriz de confusión revela el rendimiento del modelo en la detección de fuego y humo al evaluar con imágenes no vistas anteriormente. En este aspecto, el modelo es notablemente exitoso, alcanzando una precisión del **76 %** en la detección de la clase “fuego” y del **79 %** en la detección de la clase “humo”. Estos resultados son prometedores, considerando la complejidad de los patrones de estos objetos y la dificultad adicional de las variaciones en la escala térmica de las imágenes.

Además, al combinar ambos conjuntos de datos, el modelo logra realizar predicciones más precisas y generalizadas. Aunque en algunos casos, especialmente con fogatas similares a las presentes en el conjunto CORSICAN, se observa una ligera disminución en la precisión, el modelo demuestra una mayor capacidad para detectar anomalías en otros entornos. Esto indica que la diversidad en los datos de entrenamiento permite mejorar la detección en condiciones variadas, incrementando la adaptabilidad del modelo sin comprometer significativamente el rendimiento en situaciones más específicas (Figura 5.33).



Ilustración 5.33: Ejemplos de detecciones predichas por el modelo YOLOv8n entrenado con imágenes reales de CORSICAN y generadas sintéticamente con Juggernaut XL. Se observa cómo patrones de fuego reales que son grabados de manera cercana pueden ser detectados y algunas imágenes de otros entornos puede llegar a detectar humo y fuego.

Cuadro 5.4: Comparación del rendimiento de YOLOv8 en diferentes conjuntos de datos. Fuente: Datos obtenidos de los resúmenes de métricas.

Conjunto de Datos	Precisión	Recall	mAP@50	mAP@50-95
Solo Sintéticas (Juggernaut XL)	72.60 %	66.32 %	71.53 %	38.73 %
Solo CORSICAN	96.95 %	94.08 %	97.91 %	78.88 %
Sintéticas + CORSICAN	83.15 %	71.13 %	79.45 %	50.53 %

En el Cuadro 5.4 se presenta el rendimiento general del modelo YOLOv8n en cada conjunto de datos. Este análisis nos permite extraer conclusiones importantes sobre el impacto de la composición del conjunto de datos en el rendimiento de un modelo de aprendizaje profundo para detección de anomalías térmicas.

Los resultados evidencian la importancia de contar con un conjunto de datos variado para lograr un aprendizaje más efectivo. Un conjunto de datos limitado o homogéneo puede llevar a problemas de sobreajuste (*overfitting*), donde el modelo aprende patrones específicos sin generalizar adecuadamente, lo que resulta en un sesgo que afecta su desempeño en nuevos escenarios.

Además, la incorporación de datos sintéticos fue un recurso valioso en este proyecto. Los resultados demuestran que un modelo entrenado únicamente con imágenes generadas por inteligencia artificial puede, de hecho, detectar anomalías térmicas en imágenes reales. Esto subraya el potencial de las imágenes sintéticas como una alternativa poderosa en aplicaciones donde el acceso a datos reales es limitado o costoso.

Por último, la combinación de datos sintéticos y reales resultó en un modelo con un rendimiento superior y un alto potencial de generalización. Este enfoque refuerza la idea de que la generación de datos mediante modelos generativos, como los que emplean inteligencia artificial, puede ser una herramienta eficaz y rentable para fortalecer los conjuntos de datos de entrenamiento en tareas de detección, mejorando así la precisión y la adaptabilidad del modelo en situaciones diversas.

5.5.4. Optimización de Detección usando Técnicas de Postprocesamiento

Aunque el modelo entrenado logra predicciones sólidas y una alta confianza al detectar fuego y humo, en ciertas instancias tiende a generar múltiples *bounding boxes* en una misma área o a presentar detecciones con baja confianza. Para reducir estos problemas y mejorar la precisión de las detecciones, exploramos algunas técnicas de postprocesamiento. Nuestro objetivo fue evaluar si estas técnicas podían consolidar las detecciones en una sola *bounding box* por región y reducir los falsos positivos, optimizando así el rendimiento del modelo en condiciones reales.

Supresión de No Máximos (NMS)

Una técnica clave en el postprocesamiento es la **Supresión de No Máximos** (NMS, *Non-Maximum Suppression*), la cual se utiliza para evitar múltiples detecciones dentro de una misma área de interés. En términos generales, NMS selecciona la *bounding box* con la mayor probabilidad (confianza) en cada región y descarta las demás, basándose en un umbral de Intersección sobre Unión (IoU). Esto asegura que cada objeto detectado en una imagen esté representado por una única *bounding box*.

Aplicación de NMS en YOLOv8

El modelo YOLOv8 ya incorpora la técnica de Non-Maximum Suppression (NMS) en su función de predicción, utilizando por defecto un umbral de Intersección sobre Unión (IoU) de 70 % y un umbral de confianza del 25 %. Estos parámetros permiten filtrar detecciones para reducir redundancias y mejorar la precisión en las predicciones. Para optimizar los resultados, exploramos ajustes en ambos parámetros, buscando un equilibrio que mejorara las detecciones sin añadir falsos positivos o eliminar detecciones importantes.

El umbral de confianza controla el mínimo nivel de probabilidad requerido para que una detección se considere válida. Por defecto, YOLOv8 establece un 25 %, lo que significa que cualquier *bounding box* generada con una confianza inferior a este valor será descartada. Aumentar este umbral reduce la probabilidad de falsos positivos, mientras que disminuirlo permite más detecciones de baja confianza, aunque esto puede llevar a incluir detecciones erróneas.

El umbral de IoU de 70 % define cuánta superposición deben tener dos *bounding boxes* para considerarse duplicadas y fusionarse en una sola. Un valor de IoU más bajo tiende a eliminar detecciones adicionales en áreas superpuestas, lo cual ayuda a consolidar múltiples cajas en una sola para el mismo objeto. Ajustar el IoU puede permitir un mejor balance entre la precisión de detección y la reducción de duplicados.

Al modificar estos parámetros, evaluamos si la reducción de falsos positivos y la consolidación de *bounding boxes* en áreas superpuestas optimizaban el rendimiento del modelo sin comprometer la detección de fuego y humo en las imágenes de prueba.

Cuadro 5.5: Comparación de resultados con diferentes configuraciones de postprocesamiento en YOLOv8n. Cada columna representa un ejemplo de detección de fuego y humo en cinco imágenes distintas (Imagen 1 - Imagen 5) bajo diversas configuraciones de *IoU* y umbral de confianza. Observamos que reducir el *IoU* o aumentar el umbral de confianza puede ayudar a consolidar y afinar las detecciones, optimizando la precisión en escenarios específicos.

Configuración	Imagen 1	Imagen 2	Imagen 3	Imagen 4	Imagen 5
(IoU=0.7, Conf=0.25)					
(IoU=0.5, Conf=0.25)					
(IoU=0.7, Conf=0.5)					
(IoU=0.5, Conf=0.5)					

Como se observa en el Cuadro 5.5, ajustar los valores de IoU y confianza permite mejorar significativamente algunas detecciones. Probamos a modificar cada parámetro por separado y luego en combinación, comparando su efecto en la calidad de las detecciones.

En general, cambiar cada parámetro individualmente mejora el rendimiento del modelo con respecto a los valores base. Por ejemplo, en la **Imagen 1**, al reducir el IoU, se eliminan algunas cajas redundantes. Al aumentar el umbral de confianza, como en la **Imagen 5**, se logra reducir los falsos positivos: aunque la modificación de IoU no fue suficiente para eliminar una caja adicional de fuego (posiblemente por una intersección menor al 50 %), al combinarla con un nivel de confianza del 50 %, se eliminó la caja que inicialmente presentaba una confianza de solo 38 %.

Con estos resultados, concluimos que este equilibrio es el más adecuado para el modelo. Una reducción mayor en el IoU podría fusionar detecciones que deberían mantenerse separadas, y un umbral de confianza más alto, aunque aumentaría la precisión de las detecciones, también podría llevar a la pérdida de detecciones relevantes.

5.5.5. Evaluación Final del Modelo Propuesto en Datos Reales Provenientes de QAISC

Como prueba final del proyecto, evaluamos el rendimiento del modelo en un entorno real proporcionado por QAISC, con el objetivo de observar cómo responde ante un conjunto de datos auténticos y verificar su precisión en la detección, particularmente en la reducción de falsos positivos. Esta prueba fue diseñada para comprobar si el modelo generaba alertas indebidas en situaciones sin la presencia de fuego o humo.

QAISC nos proporcionó un conjunto de imágenes capturadas con la cámara térmica PTZ en la zona de PLOCAN y en el área del aerogenerador. Estas capturas abarcan un periodo de 7 días y fueron realizadas en intervalos de 10 minutos, totalizando **1832** imágenes.

El proceso de prueba consistió en realizar inferencias en cada una de las imágenes con el modelo configurado a un umbral de **IoU** de 0.5 y una confianza de 0.5, registrando todas las detecciones de fuego y humo en un fichero. Al analizar los resultados, observamos que el modelo no generó falsos positivos en ninguna de las imágenes sin presencia de fuego o humo. Para verificar aún más su precisión, incluimos intencionalmente dos imágenes con anomalías térmicas (presencia de fuego), y el modelo identificó correctamente el fuego y el humo en estas imágenes específicas.

Este resultado refuerza la capacidad del modelo para operar en un entorno real, proporcionando detecciones fiables sin generar falsos positivos, un aspecto fundamental para su posible aplicación en sistemas de seguridad para la detección de anomalías térmicas.

Capítulo 6

Conclusiones y trabajo futuro

El presente Trabajo Fin de Grado se ha traducido en tareas de investigación y desarrollo con el objetivo de obtener un modelo de aprendizaje profundo para la detección de anomalías térmicas en diferentes entornos, enfocándose especialmente en áreas marítimas y portuarias gracias a la colaboración con QAISC (*Qualitas Artificial Intelligence and Science*) y PLOCAN (Plataforma Oceánica de Canarias). Esta colaboración no solo orientó el proyecto hacia un entorno específico, sino que también permitió un aprendizaje práctico sobre el funcionamiento de cámaras PTZ térmico-ópticas y el procesamiento de imágenes capturadas para análisis de seguridad.

Durante el proyecto, se abordaron múltiples objetivos que presentaban desafíos complejos, permitiéndonos profundizar en áreas como visión por computador y aprendizaje profundo. Esto incluyó el uso de metodologías ágiles, como el enfoque KANBAN para priorizar tareas, así como una metodología de desarrollo en fases que nos permitió estructurar el proyecto eficientemente. Además, aplicamos conocimientos técnicos adquiridos durante el Grado en Ingeniería Informática, incluyendo el uso de entornos Linux, Docker, y lenguajes de programación como Python, junto con bibliotecas y frameworks especializados como NumPy, OpenCV, Keras y PyTorch.

Para el entrenamiento de los modelos, recopilamos una amplia base de imágenes en infrarrojo y exploramos la generación de datos sintéticos mediante redes GAN, incluyendo técnicas como Juggernaut XL y herramientas de acondicionamiento como IP Adapter y ControlNet, que facilitaron la creación de un dataset amplio y variado. La generación de imágenes sintéticas resultó ser una solución eficiente para enriquecer el conjunto de datos, mejorando notablemente el rendimiento del modelo en comparación con un entrenamiento basado únicamente en imágenes reales.

A nivel técnico, exploramos diferentes arquitecturas de modelos de aprendizaje profundo para la detección de incendios. Utilizamos el modelo Xception para la clasificación y U-Net para la segmentación semántica, ambos evaluados con imágenes reales en RGB e IR del conjunto de datos FLAME. Para la detección de objetos, implementamos YOLOv8 y aquí sí incorporamos imágenes sintéticas generadas mediante IA para enriquecer el conjunto de datos, logrando un rendimiento robusto. La inclusión de datos sintéticos en YOLOv8, en

particular, mejoró la precisión cuando se combinaron con datos reales. Además, el modelo YOLOv8 fue optimizado mediante ajustes en el umbral de confianza y el parámetro de Intersección sobre la Unión (IoU), lo que permitió mejorar la precisión y reducir los falsos positivos en las detecciones.

A pesar de los buenos resultados obtenidos, el modelo final tiene margen de mejora, especialmente debido a los desafíos inherentes al procesamiento de imágenes térmicas. Para trabajos futuros, se pueden considerar varias líneas de investigación:

1. **Generación avanzada de imágenes sintéticas:** Continuar explorando la generación de datos mediante modelos de texto a imagen, probando otros modelos hiperrealistas e incluso entrenando uno propio. Esto podría permitir la creación de imágenes más específicas y adaptadas a la escala de color térmica *WhiteHot*, mejorando la variabilidad de los datos de entrenamiento.
2. **Mejora de la generación de datos mediante GANs:** Con una mayor cantidad de imágenes IR de incendios, aunque sean sintéticas, sería posible mejorar los resultados obtenidos con redes GAN, proporcionando al modelo una base de datos más rica y diversificada.
3. **Captura de datos reales controlados:** Obtener imágenes de incendios controlados capturadas con cámaras IR permitiría afinar aún más el modelo y comprender mejor su comportamiento en entornos reales, ajustando los umbrales de temperatura y mejorando la precisión en condiciones de campo.
4. **Estudios de ablación:** Realizar estudios de ablación modificando parámetros y configuraciones del modelo, así como explorando ajustes en la arquitectura, para evaluar su impacto en la precisión de detección y reducir posibles sesgos.

En conclusión, este proyecto consolidó y amplió nuestros conocimientos en inteligencia artificial y visión por computador, resultando en un modelo de detección de anomalías térmicas funcional y preciso, con un gran potencial de aplicación en entornos reales para mejorar la seguridad, reducir riesgos y costos. Asimismo, el trabajo realizado establece una base sólida para futuras investigaciones y desarrollos en generación de datos sintéticos y en el perfeccionamiento de sistemas de detección de anomalías térmicas asistidos por IA.

Glosario

U-Net Arquitectura de red neuronal utilizada principalmente en segmentación de imágenes, que asigna etiquetas a nivel de píxel para identificar áreas específicas en una imagen. 17

Xception (Extreme Inception) Arquitectura de red neuronal convolucional basada en la idea de convoluciones separables en profundidad, utilizada principalmente en tareas de clasificación de imágenes y otras aplicaciones de visión por computador. 11

anomalías térmicas Variación inesperada en la temperatura de un área, que puede indicar eventos como incendios o sobrecalentamientos. 1

cámara bi-espectro Dispositivo que combina información de dos espectros diferentes, como el espectro térmico y el óptico, permitiendo un análisis más completo de las escenas capturadas. 2

detección de objetos Técnica de visión por computador para identificar y localizar instancias de objetos dentro de una imagen mediante bounding boxes. 8

generalización Capacidad de un modelo de aprendizaje automático para desempeñarse bien con datos no vistos previamente, más allá de los utilizados en el entrenamiento. 17

redes generativas adversarias (GANs) Arquitectura de aprendizaje profundo compuesta por dos redes neuronales (un generador y un discriminador) que compiten entre sí, utilizada principalmente para generar datos nuevos y realistas a partir de datos existentes. 9

redes neuronales convolucionales (CNN) Arquitectura de red neuronal diseñada para procesar datos con estructura en forma de cuadrícula, como imágenes. 8

termogramas Representación visual de la radiación infrarroja emitida por los objetos en forma de imagen térmica, utilizada para identificar diferencias de temperatura. 1

YOLO (*You Only Look Once*) Modelo de aprendizaje profundo especializado en la detección rápida y precisa de objetos dentro de imágenes. 8

Bibliografía

- [1] Anderson, D. J. (2010). *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press.
- [2] Authors, D. (2023a). Deepdepth: Enhancing depth-aware text-to-image generations.
- [3] Authors, I.-A. (2023b). Ip-adapter: Image prompting adapter for text-to-image diffusion models.
- [4] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.
- [5] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1251–1258.
- [6] Dalhousie (2023). Marine image thermal dataset.
- [7] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE.
- [8] Drucker, P. F. (1954). *The Practice of Management*. Harper & Row.
- [9] Escuela de Ingeniería Informática, Universidad de Las Palmas de Gran Canaria (2019). Grado en ingeniería informática - memoria del plan de estudios.
- [10] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., and Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118.
- [11] Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338.
- [12] Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J. (2021). YoloX: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*.

- [13] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.
- [14] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.
- [15] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.
- [16] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.
- [17] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778.
- [18] Huang, X., Liu, M.-Y., Belongie, S., and Kautz, J. (2018). Multimodal unsupervised image-to-image translation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 172–189.
- [19] Indeed (2022). What is management by objectives (mbo) and how does it work?
- [20] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.
- [21] Jin, Y., Zhang, Y., Li, W., Lu, Y., and Pan, C. (2019). Deep learning-based structural health monitoring. *Journal of Sensors*, 2019:4578091.
- [22] Jocher, G., Chaurasia, A., and Qiu, J. (2023). Ultralytics yolov8.
- [23] Kanbanize (2021). The history of kanban: How toyota changed manufacturing.
- [24] Kaplan, R. S. and Norton, D. P. (1996). *The Balanced Scorecard: Translating Strategy into Action*. Harvard Business School Press.
- [25] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- [26] Ladas, C. (2009). *Scrumban: Essays on Kanban Systems for Lean Software Development*. Modus Cooperandi Press.
- [27] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- [28] Lee, D.-G. and Kim, A. (2023). Edge-guided multi-domain rgb-to-tir image translation for training vision tasks with challenging labels.

- [29] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2015). Microsoft coco: Common objects in context.
- [30] Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J. (2018). Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [31] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37.
- [32] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.
- [33] Microsoft (2024). Visual studio code.
- [34] Minsky, M. and Papert, S. A. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press.
- [35] Nirgudkar, S., DeFilippo, M., Sacarny, M., Benjamin, M., and Robinette, P. (2023). Massmind: Massachusetts marine infrared dataset. *The International Journal of Robotics Research*.
- [36] Ohno, T. (1988). *Toyota Production System: Beyond Large-Scale Production*. Productivity Press.
- [37] OpenCV (2022). Computer vision annotation tool (cvat): Comprehensive guide.
- [38] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*.
- [39] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788.
- [40] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.
- [41] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695.
- [42] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

- [43] Ros, G., Sellart, L., Materzynska, J., Vazquez, D., and Lopez, A. M. (2016). The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [44] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408.
- [45] Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Appalaraju, S., Kannan, A., and Norouzi, M. (2022). Imagen: Text-to-image diffusion models. *Proceedings of the International Conference on Machine Learning (ICML)*.
- [46] Shamsoshoara, A., Afghah, F., Razi, A., Zheng, L., Fulé, P. Z., and Blasch, E. (2021). Aerial imagery pile burn detection using deep learning: the flame dataset. *Computer Networks*, page 108001.
- [47] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
- [48] Suo, J., Wang, T., Zhang, X., Chen, H., Zhou, W., and Shi, W. (2023). Hit-uav: A high-altitude infrared thermal dataset for unmanned aerial vehicle-based object detection. *Scientific Data*, 10:227.
- [49] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9.
- [50] Trello (2022). How to use kanban in trello: A beginner’s guide.
- [51] Wang, C.-Y., Liao, H.-Y. M., Chen, Y.-H., Chen, P.-Y., Hsieh, J.-W., and Yeh, I.-H. (2020). Cspnet: A new backbone that can enhance learning capability of cnn. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.
- [52] Womack, J. P., Jones, D. T., and Roos, D. (1990). *The Machine That Changed the World*. Simon & Schuster.
- [53] Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. (2017). Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5907–5915.
- [54] Zhang, L. and Agrawala, M. (2023). Controlnet: Adding conditional control to text-to-image diffusion models.
- [55] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.