

# Técnicas de *Deep-Learning* para la detección de personas en entornos de baja visibilidad fusionando imágenes visuales y de infrarrojos

Máster en Ingeniería Electrónica, Robótica y Automática

Realizado por:  
José García Gómez,  
José Luis Montes Reyes,  
Pablo Díaz Cotrino,  
María Leal Suárez,  
Gloria León Lago

# Índice

|  |           |
|--|-----------|
| <b>Índice.....</b>   | <b>2</b>  |
| <b>Introducción.....</b>   | <b>3</b>  |
| <b>Estado del arte.....</b>  | <b>4</b>  |
| <b>Dataset.....</b>  | <b>7</b>  |
| Elección del dataset.....  | 7         |
| Datasets considerados.....   | 7         |
| Estructura.....  | 7         |
| Anotaciones.....   | 8         |
| <b>Fusión de imágenes.....</b>                                       | <b>9</b>  |
| <b>Arquitecturas de detectores.....</b>                              | <b>11</b> |
| Arquitecturas de detectores consideradas y criterio de elección..... | 11        |
| Faster R-CNN.....  | 11        |
| DETR.....  | 11        |
| Comparativa teórica y resumen de la elección.....                    | 12        |
| <b>Implementación y entorno.....</b>                                 | <b>13</b> |
| <b>Entrenamiento y evaluación.....</b>                               | <b>14</b> |
| Hiperparámetros utilizados.....                                      | 14        |
| Métricas de evaluación.....  | 14        |
| Resultados del entrenamiento.....                                    | 14        |
| Coste computacional y uso de GPU.....                                | 17        |
| <b>Detección y evaluación.....</b>                                   | <b>18</b> |
| Resultados obtenidos.....  | 18        |
| Histograma de scores y elección del umbral.....                      | 20        |
| Rendimiento computacional en predicción.....                         | 21        |
| <b>Comparación de entrenamiento con DETR.....</b>                    | <b>22</b> |
| Resultados y comparación con Faster R-CNN.....                       | 22        |
| Coste computacional: tiempo y memoria.....                           | 24        |
| Conclusión de la comparación.....                                    | 24        |
| <b>Comparación de detección con DETR.....</b>                        | <b>26</b> |
| Resultados en test y comparación con Faster R-CNN.....               | 26        |
| Histograma de scores.....  | 28        |
| Rendimiento computacional en predicción.....                         | 28        |
| Conclusión de la comparación.....                                    | 29        |
| <b>Conclusiones y trabajo futuro.....</b>                            | <b>30</b> |
| <b>Referencias.....</b>  | <b>32</b> |

# Introducción

En la actualidad, la capacidad de percibir el entorno con detalle de una manera fiable es un requisito fundamental para garantizar operaciones eficientes en robótica móvil y vehículos autónomos. En este contexto, la visión artificial se ha convertido en una pieza clave, especialmente desde la integración del Deep Learning, que ha permitido mejorar de manera notable tareas como la detección y seguimiento de objetos.

Aun así, los sistemas tradicionales basados en cámaras de espectro visible (RGB) siguen teniendo limitaciones importantes cuando las condiciones no son favorables. En escenas con poca iluminación, niebla, humo o polvo, la calidad de la imagen se degrada y, con ella, la fiabilidad del sistema. Esto supone un problema directo en aplicaciones donde la detección debe funcionar de forma consistente incluso en escenarios adversos.

Frente a esto, la imagen infrarroja (IR) aparece como una alternativa interesante, ya que captura la radiación térmica emitida por los objetos y no depende tanto de la iluminación externa. Para la detección de personas resulta especialmente útil, porque la firma térmica humana suele ser distingible incluso en oscuridad total. Sin embargo, el IR también tiene sus propias limitaciones: pierde información de textura y color, y en determinadas escenas puede ser difícil separar objetos con temperaturas similares al fondo o con poco contraste térmico.

La motivación de este proyecto nace de la necesidad de desarrollar un sistema de percepción robusto combinando información RGB e IR, de forma que una modalidad compense las debilidades de la otra. La idea es que el sistema pueda mantener un rendimiento estable en condiciones de baja visibilidad, donde una cámara convencional por sí sola no sería suficiente.

Este proyecto tiene como objetivo principal el diseño y la implementación de un sistema de detección de personas basado en aprendizaje profundo que aproveche la fusión de imágenes visibles e infrarrojas.

Para alcanzar el objetivo se ha llevado a cabo una profunda investigación de métodos de fusión y la selección de los datasets más adecuados para este proceso, datasets que contuviesen pares de imágenes RGB e IR alineados para la detección de personas. Se ha evaluado y adaptado el dataset LLVIP (Low-Light Visible-Infrared Paired) para su uso en arquitecturas de aprendizaje profundo, implementando algoritmos de alineación y normalización de datos.

En concreto, se han desarrollado técnicas de fusión mediante el apilado de canales, generando entradas multimodales de 4 canales y la modificación de modelos de detección de objetos evaluando arquitecturas como Faster R-CNN o DETR para admitir entradas multiespectrales y entrenarlos bajo las condiciones específicas del proyecto.

Finalmente, se han llevado a cabo varios entrenamientos con las arquitecturas mencionadas, y realizado una evaluación detallada del rendimiento aportado por los modelos obtenidos.

# Estado del arte

A continuación, se llevará a cabo un estado del arte de distintas técnicas de fusión de imágenes visuales y de infrarrojos y sus distintas aplicaciones en la detección y seguimiento tanto de personas como de otros objetos, además de algún método de detección en condiciones de baja visibilidad. Los distintos artículos que se describirán a continuación permitirán concebir una idea general de la variedad de los posibles métodos de fusión y su conveniencia en la detección de elementos a partir de un par de imágenes visible e infrarroja, por lo que su análisis servirá como punto de partida para la realización de este proyecto.

En el primero de los artículos que se van a describir a continuación [1], se han llevado a cabo diferentes métodos de fusión de pares de imágenes visible e infrarroja aplicados a un problema de seguimiento de objetivos en un sistema de video-vigilancia. Los métodos de fusión que se evalúan en dicho artículo pueden dividirse en dos grupos principales: métodos de combinación simple y métodos basados en estructuras en pirámide. Los métodos que se encuentran dentro del primero de los grupos consisten en una fusión en la que cada píxel toma un valor de intensidad en función del valor que tengan los píxeles correspondientes en las imágenes visible e infrarroja: el nuevo valor de cada píxel puede ser dado por la media de ambas imágenes, la media ponderada según su varianza o el valor máximo o mínimo píxel a píxel de las intensidades de ambas imágenes.

Los métodos pertenecientes al segundo grupo consisten en la descomposición multirresolución de las imágenes visibles e infrarrojas basada en pirámides. Entre los métodos pertenecientes a este grupo se encuentran el método de fusión por pirámide Laplaciana (que descompone la imagen en componentes de baja y alta frecuencia en las distintas capas de la pirámide, cada una con menor resolución que la anterior, combina capa a capa las imágenes visible e infrarroja y deshace la descomposición), el FSD (similar al anterior, pero omitiendo el proceso de filtrado al reconstruir la imagen), la fusión por pirámide de contraste o de ratio de paso bajo o ROLP (cuya principal diferencia con la Laplaciana es la forma en la que computan las capas de alta frecuencia de su pirámide, sustituyendo la resta de componentes de baja frecuencia por un ratio menos uno o por un ratio respectivamente).

Además, alejándose de las pirámides basadas en la Laplaciana, se encuentran la pirámide de gradiente (que calcula los gradientes en las imágenes en las direcciones horizontal, vertical y diagonales, preservando la información de orientación de los objetos), la pirámide morfológica (que encadena en cada capa operaciones de apertura y cierre, lo que lleva a altos costos computacionales) y las pirámides basadas en las transformadas *wavelet* (que separan la imagen en aproximaciones y detalles) como la transformada *wavelet* discreta DWT, la SIDWT (DWT invariante al desplazamiento, mediante la *wavelet* de Haar, lo que mitiga el *aliasing*) o la DT-CWT (la transformada *wavelet* compleja con doble árbol).

Este artículo combina cada uno de los métodos de fusión propuestos anteriormente con un mismo sistema de seguimiento por sustracción de fondo, FPSS (*Force Protection Surveillance System*). Este hace uso de cuatro memorias intermedias FIFO (*buffers*) y una máscara de estabilidad, que permite al algoritmo identificar píxeles inestables (o móviles).

Este sistema lleva a cabo un cálculo de una *Imagen de Producto de Diferencias* como el producto píxel a píxel de las diferencias entre la imagen y el modelo que ha desarrollado del fondo, para así resaltar independientemente del brillo los objetos móviles. Sobre esta imagen calculada se estiman los centroides de los objetos móviles, además de predecir su movimiento a partir de su posición, velocidad y tamaño en imágenes anteriores.

Tras evaluar cada uno de los cuatro métodos de combinación simple y los nueve basados en estructuras en pirámide, cuyas salidas serían las entradas del algoritmo de seguimiento FPSS, el artículo reporta resultados peores al utilizar los métodos de combinación simple que con la mayoría de métodos basados en pirámide. De hecho, se muestra que estos dan resultados incluso peores que si se emplea únicamente la imagen infrarroja. De entre los métodos basados en pirámides, los métodos FSD, la pirámide de gradiente, la DWT y la pirámide morfológica también dan resultados similares o peores que usar únicamente la imagen infrarroja (siendo este último método el que peores resultados da con diferencia). Los métodos basados en pirámides restantes dan resultados bastante mejores que los anteriores, siendo los métodos de pirámide ROLP y de contraste y la DT-CWT los que mejores resultados reportan de entre ellos.

El segundo de los artículos que se han analizado lleva a cabo una fusión de imágenes visuales e infrarrojas de forma completamente diferente al comentado anteriormente. En él, dicha fusión no se lleva a cabo mediante reglas estáticas, sino que hace uso de una red neuronal de múltiples etapas, la cuál toma como entradas, por separado, la imagen visual, la imagen en infrarrojos y la imagen resultante del apilamiento de ambas; esta red ha sido entrenada con un aprendizaje no supervisado, evitando así la necesidad de contar previamente con una imagen fusionada “perfecta” con la que comparar los resultados.

El modelo desarrollado para la fusión de las imágenes se descompone en tres etapas: codificación, cuyo objetivo principal consiste en la extracción de características de cada imagen por separado (visual e infrarroja, de las que se extraen características específicas, y el apilado de ambas, de la cual se extraen características comunes) mediante bloques densos, fusión, que lleva cabo una suma de las características de la imagen visual y la imagen infrarroja, cuyo resultado se utiliza para hacer una media ponderada con las características de la imagen de canales apilados, y decodificación que, mediante varias capas convolucionales, reconstruye una imagen fusionada a partir de las múltiples características de las imágenes originales. La función de pérdida utilizada en este artículo es una función de pérdida de píxel (mediante el error cuadrático cometido) combinada con un índice de similitud estructural, SSIM, que permite medir cuánto se asemejan las imágenes resultantes a las imágenes de partida.

Para evaluar la calidad del modelo propuesto por este artículo, se han comparado métricas obtenidas por este (como la entropía de información, la desviación estándar, el gradiente medio o la información mutua) al afrontar algunos datasets públicos, como OSU, LITIV o VAIS con otros métodos como FVP, CSR, GTM o TSD. Los resultados obtenidos muestran una gran capacidad de retención de información importante y de resaltar objetos térmicos sin que ello implique una pérdida en la nitidez de la imagen o una pérdida de las características visuales.

Finalmente, el último de los artículos descritos en el estado del arte se centra principalmente en la segunda fase del proyecto: la detección. En este artículo [3] se hace uso de una adaptación basada en ConvNet del modelo Faster R-CNN para la detección de peatones mediante imágenes visibles e infrarrojas, lo que se alinea con los objetivos de este proyecto. En primer lugar, se realizaría un entrenamiento separado con dos redes, de tal forma que cada una de ellas recibiera bien la imagen visual bien la imagen en infrarrojos. Al hacerlo, se pudo comprobar que las características obtenidas por cada red eran complementarias entre sí, lo que volvía prometedora la idea de alimentar una red de una estructura similar simultáneamente con imágenes visuales e infrarrojas.

En este artículo se propusieron cuatro alternativas de fusión para su red de detección basadas en Faster R-CNN: fusión temprana (se concatenan las características extraídas del par de imágenes inmediatamente después de la primera capa convolucional), fusión intermedia (la fusión de características se produce tras la cuarta capa convolucional), fusión tardía (producida tras la última capa *fully-connected*) y fusión de puntuación (la fusión se produce a nivel de decisión, no de característica, además de introducir en la red de infrarrojo la salida de la red de color y viceversa).

Estos cuatro métodos se han comparado entre sí, además de usar como *baseline* su red adaptada de Faster R-CNN, la cual demostró mejores resultados que otros métodos como HOG, DBN-Mut, LDCF o Katamari. Al comparar todos estos métodos tanto con el *baseline* como entre sí, se muestra que la fusión intermedia es la que obtiene mejores resultados en todas las circunstancias analizadas con el dataset KAIST, seguida por la fusión temprana en imágenes diurnas y por la fusión de puntuación en imágenes nocturnas (aunque esta última obtiene resultados peores que varios de los otros métodos en el resto de casos).

# Dataset

## Elección del dataset

Para el desarrollo de este proyecto se seleccionó el LLVIP [4] (*Low-Light Visible-Infrared Paired Dataset*) como conjunto de datos principal. La decisión tomada de utilizar este dataset se fundamenta en los objetivos del proyecto, centrados en la detección robusta de personas en condiciones de baja visibilidad mediante la fusión de información visible e infrarroja.

LLVIP está específicamente diseñado para escenarios de muy baja iluminación, proporcionando pares de imágenes visibles e infrarrojas térmicas capturadas de forma simultánea y alineadas especialmente. Este dataset contiene un total de 15488 pares de imágenes, es decir, 30976 imágenes en total, y está orientado principalmente a escenas urbanas nocturnas o con iluminación muy degradada.

Este enfoque resulta adecuado para el objetivo del proyecto, dado que en muchas escenas visibles del dataset la información aportada por la imagen RGB es limitada, mientras que la modalidad infrarroja permite identificar claramente la presencia de personas.

## Datasets considerados

Durante la fase inicial del proyecto se analizaron otros datasets visibles-infrarrojos, entre ellos *KAIST Multispectral Pedestrian Dataset*, CVC-14 y *FLIR Thermal Dataset*.

El dataset KAIST [5], aunque es un referente en detección multiespectral de peatones y cuenta con un gran número de imágenes y anotaciones, está orientado principalmente a escenarios de conducción urbana desde vehículo, con escenas nocturnas que, en general, mantienen un nivel de iluminación superior al considerado en este proyecto. CVC-14 presenta una problemática similar, ya que sus escenas nocturnas no representan condiciones de iluminación suficientemente degradadas, y el peatón suele ser fácilmente identificable incluso en la modalidad visible. Por su parte, el dataset FLIR [6], aunque presenta un gran número de imágenes anotadas, no ofrece pares visible-infrarrojo perfectamente alineados, lo que dificulta su uso en estrategias de fusión.

Por estos motivos, y tras comparar las características de los distintos conjuntos de datos, se decidió utilizar LLVIP como dataset principal, al ser el que mejor se ajusta a los requisitos del proyecto.

## Estructura

El dataset LLVIP se organiza de forma estructurada para facilitar el uso conjunto de los dos tipos de imágenes, así como la asociación directa con las anotaciones correspondientes. La organización del conjunto de datos mantiene una correspondencia uno a uno entre imágenes y etiquetas, lo que resulta adecuado para tareas de detección multimodal.

El dataset se divide de la siguiente forma:

- Conjunto de entrenamiento: 12025 pares de imágenes visible-infrarrojo.
- Conjunto de prueba: 3463 pares de imágenes visible-infrarrojo.
- Anotaciones: 15488 anotaciones en formato Pascal VOC en forma de ficheros XML.

Cada fichero de anotación XML utiliza el mismo nombre base que la imagen visible y la imagen infrarroja correspondientes, lo que permite asociar de manera directa cada par de imágenes con sus etiquetas. Esta igualdad de nombres garantiza que las anotaciones sean aplicables tanto a la imagen visible como a la infrarroja sin inconsistencias.

## Anotaciones

LLVIP proporciona anotaciones de detección de peatones en formato Pascal VOC, almacenadas en ficheros XML, uno por imagen. Esto significa que una misma anotación se aplica tanto a la imagen visible como a la infrarroja, garantizando la coherencia del *ground truth* en tareas multimodales.

Cada archivo XML incluye la siguiente información relevante:

- Metadatos de la imagen, como el nombre del archivo y el tamaño de la imagen.
- Listado de objetos anotados, donde cada objeto corresponde a una persona presente en la escena.
- Para cada objeto se especifican:
  - la clase, que en este dataset corresponde a person,
  - información adicional como *pose*, *truncated* y *difficult*,
  - y el *bounding box* definido mediante las coordenadas absolutas de sus vértices en píxeles.

Una misma imagen puede contener varias instancias de la clase *person*, y todas ellas se mantienen como parte del *ground truth*. Las coordenadas de los *bounding boxes* se expresan en el sistema de referencia de la imagen y son directamente aplicables a ambas modalidades, ya que las imágenes visible e infrarroja comparten las mismas dimensiones y alineación.

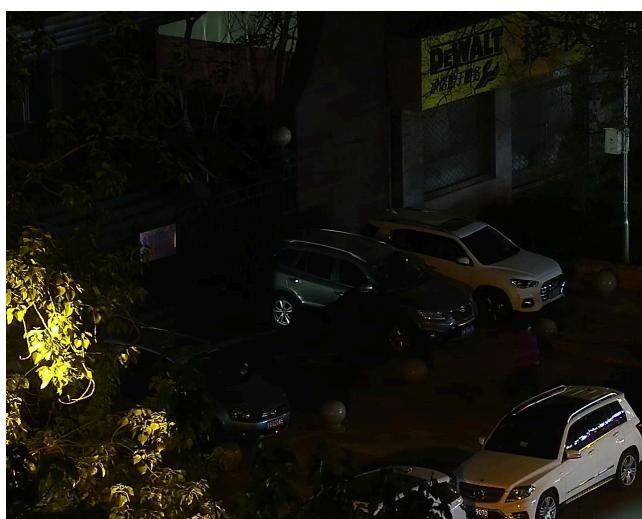


Ilustración 1. Imagen visible del dataset LLVIP.



Ilustración 2. Imagen infrarroja del dataset LLVIP.

# Fusión de imágenes

En el estado del arte se han analizado algunos artículos que emplean métodos de fusión diferentes que podían aportar ideas sobre cómo llevar a cabo la fusión de imágenes visuales e infrarrojas de forma que permitiera la detección de personas en entornos de baja visibilidad. Durante la realización de este proyecto han surgido diversas posibilidades para llevar a cabo dicha fusión, las cuales se explicarán a continuación.

En primer lugar, se planteó desarrollar una fusión de forma similar a como se realizó en [1]; es decir, a partir del dataset a utilizar, que incluye pares de imágenes visibles e infrarrojas, crear una copia de dicho dataset en la que, en lugar de estas parejas de imágenes, se encontrasen las imágenes ya fusionadas. Para ello, se realizaría una conversión de la imagen visible de RGB a escala de grises para, a continuación, llevar a cabo la fusión mediante algunos métodos estáticos como los mencionados en el artículo: máximo, mínimo o promedio de las imágenes o métodos basados en pirámides como la Laplaciana o ROLP, o basados en la transformada de *wavelet* [1].

Este método tiene la cualidad de que la red de detección ya recibiría la imagen fusionada monocromática, lo cual llevaría a una carga computacional mucho menor tanto durante el entrenamiento como durante la evaluación y a permitir a la red neuronal utilizada centrarse únicamente en la detección de personas. Sin embargo, esto supone llevar a cabo un preprocesamiento del dataset, con una carga computacional dependiente del método de fusión utilizado (a mayor calidad de la imagen resultante del método, mayor será su coste computacional), además de que, al convertir la imagen visual a escala de grises y realizar la pre-fusión, existe el riesgo de perder características de las imágenes originales que podrían resultar importantes para la detección.

La segunda de las posibilidades planteadas, y la cual sería la que se llevaría a cabo finalmente, consistiría en ajustar la etapa de entrada de la red neuronal utilizada para admitir no uno, sino cuatro canales, provenientes tanto de la imagen visual como de la infrarroja: RGB+Ir; es decir, fusionar por apilamiento de canales. De esta forma, se proporcionan ambas imágenes de forma íntegra a la red, evitando así el preprocesamiento necesario con el otro método y la posibilidad de perder características importantes debido a esto.

Es por ello que, finalmente, esta sería la decisión tomada respecto a la fusión de las imágenes, para asegurar que la etapa de detección contase con toda la información disponible y no con la filtrada tras implementar un método de fusión estático, filosofía que se alinea con la adoptada en [2] al confiar la fusión de las imágenes a la propia red. Sin embargo, este método cuenta también con algunos inconvenientes a tener en cuenta; por ejemplo, la red debe ser capaz tanto de detectar personas en una imagen más compleja (con cuatro canales en lugar de uno como la opción anterior), como de llevar a cabo de forma interna la fusión de las características dadas por cada canal y aprender cómo combinarlas, dando lugar a un mayor coste computacional tanto durante su entrenamiento como durante la inferencia y un mayor volumen de parámetros de dicha red, sobre todo en la capa de entrada a la red. Cabe señalar que, si bien en [3] se habla de fusión temprana,

esta se produce tras la primera extracción de características, mientras que la que se propone en este proyecto es a nivel de datos, no de características.

# Arquitecturas de detectores

## Arquitecturas de detectores consideradas y criterio de elección

Para la detección de personas se valoraron tres enfoques: YOLO, Faster R-CNN y DETR. La elección se hizo pensando en el objetivo del proyecto: maximizar la fiabilidad en baja visibilidad, aprovechando la información conjunta de visible e infrarrojo. En este trabajo no se exige tiempo real, así que la prioridad es precisión y robustez, aunque el modelo sea más pesado.

YOLO se descartó principalmente por ese motivo. Es un detector muy competitivo, pero su mayor ventaja es la velocidad, y aquí no compensa frente a arquitecturas que suelen rendir mejor cuando se busca exprimir la calidad de la detección y reducir el número de falsos positivos en escenas difíciles.

Con ese criterio se seleccionó Faster R-CNN como modelo principal, por ser una opción consolidada y estable cuando se busca precisión. Además, se incluyó DETR como alternativa para comparar, ya que representa un enfoque distinto (más global y *end-to-end*) que puede dar buenos resultados cuando el contexto de la escena es importante.

Finalmente, se trabaja con LLVIP, con un preprocesado previo mínimo: las muestras se transforman a una entrada fusionada en 4 canales, combinando RGB e infrarrojo en un único tensor para que el modelo aprenda a integrar ambas fuentes desde el inicio.

## Faster R-CNN

Faster R-CNN es un detector de dos etapas. Primero localiza regiones candidatas donde podría haber un objeto y después clasifica y ajusta con más precisión los bounding boxes finales. Este planteamiento suele ser una ventaja cuando la escena es compleja o la señal visual es pobre, ya que separa la búsqueda de candidatos del refinado final.

La calidad del detector depende mucho del backbone, que es el extractor de características. En detección es habitual usar backbones tipo ResNet, y añadir FPN para trabajar bien a distintas escalas, algo útil cuando el tamaño aparente de la persona cambia (distancias distintas, occlusiones, recortes). En este proyecto, el backbone es especialmente relevante al no ser la entrada RGB estándar, sino una entrada fusionada con IR: el extractor debe aprender una representación común que aproveche la información complementaria entre canales.

## DETR

DETR aborda la detección de otra manera. En lugar de generar muchas propuestas y filtrarlas, plantea una predicción *end-to-end* basada en transformers, donde el modelo produce directamente un conjunto de detecciones finales. Usa un backbone (normalmente CNN) para extraer características y un transformer para incorporar contexto global en la decisión.

Esto lo hace interesante para comparar: cuando la evidencia local es débil, el contexto puede ayudar a decidir mejor. A cambio, suele ser un modelo más exigente de entrenar y puede necesitar más cuidado para converger bien, especialmente en casos complicados (como objetos pequeños o contornos poco definidos).

## Comparativa teórica y resumen de la elección

| Arquitectura / Característica        | YOLO  | Faster R-CNN                                | DETR  |
|--------------------------------------|---|---|---|
| <b>Enfoque</b>                       | One-stage (predice todo en una pasada)  | Two-stage (propuestas refinadas) +          | End-to-end con transformer (predicción como conjunto)             |
| <b>Nº de etapas</b>                  | 1   | 2   | 1 (sin propuestas clásicas)                                       |
| <b>Backbone típico</b>               | CNN (según versión)   | CNN (ej: ResNet) + FPN habitual             | CNN + Transformer (encoder/decoder)                               |
| <b>Velocidad de inferencia</b>       | Muy alta  | Media-baja                                  | Media   |
| <b>Tiempo/coste de entrenamiento</b> | Medio (suele converger rápido)  | Medio-alto                                  | Alto (más exigente)   |
| <b>Fortalezas</b>                    | Tiempo real, pipeline simple  | Precisión y estabilidad, buena localización | Contexto global, enfoque moderno, menos heurísticas clásicas      |
| <b>Debilidades</b>                   | Menos adecuado si prima precisión máxima; puede sufrir en escenas muy difíciles | Más lento y pesado                          | Entrenamiento delicado; depende más de configuración y resolución |
| <b>Encaje en este proyecto</b>       | Bajo (no se necesita tiempo real)   | Alto (prioridad: robustez/precisión)        | Medio-alto (útil para comparar)                                   |

Tabla 1. Comparativa teórica entre arquitectura de detector YOLO vs Faster-RCNN vs DETR.

Por todo lo explicado, se elige Faster R-CNN como referencia principal por su estabilidad y por ser una opción fuerte cuando se prioriza precisión. DETR se incluye para medir si el enfoque global basado en transformers aporta mejoras reales en este escenario. YOLO se descarta porque el proyecto no está limitado por tiempo de inferencia y se busca maximizar la calidad de detección.

# Implementación y entorno

Para poder entrenar modelos de detección con un dataset de este tamaño, era necesario disponer de un entorno con capacidad de cómputo suficiente. En LLVIP se trabajan pares visible-infrarrojo y el volumen total de datos es elevado, lo que hace que entrenar en GPUs de portátil sea lento y poco práctico para iterar con facilidad.

Por este motivo, los entrenamientos se consiguieron realizar sobre una NVIDIA Jetson AGX Orin (32 GB). Esta plataforma ofrece una GPU mucho más adecuada para cargas de Deep Learning sostenidas y, sobre todo, suficiente memoria para manejar modelos de detección y lotes de datos sin estar constantemente limitados por recursos. Esto permitió reducir tiempos de entrenamiento y repetir experimentos de forma más razonable.

A nivel de software, se optó por trabajar dentro de un contenedor Docker para aislar dependencias y evitar problemas de compatibilidad. En Jetson no es recomendable usar imágenes genéricas de GPU pensadas para PCs, así que se empleó una imagen específica para la plataforma: *dustynv/l4t-pytorch:r36.4.0*, que ya integra una versión de PyTorch adaptada al ecosistema de Jetson y facilita que el entorno sea estable. De esta forma, el proyecto resulta más reproducible: mismo sistema base, mismas librerías y menos diferencias entre ejecuciones por usar distintas máquinas.

En cuanto a librerías, el núcleo del trabajo se apoyó en PyTorch y librerías habituales del ecosistema: componentes de visión (por ejemplo, para los detectores basados en CNN), librerías para modelos tipo transformer (para el caso de DETR), y utilidades comunes para carga y procesado de imágenes, operaciones numéricas y generación de gráficas.

Un punto importante de la implementación fue adaptar las arquitecturas para trabajar con la entrada fusionada. Como el enfoque elegido consiste en apilar canales y pasar a una entrada de 4 canales (RGB + IR), fue necesario ajustar la capa inicial de los modelos, normalmente diseñada para imágenes RGB de 3 canales. En la práctica, esto permite que el extractor de características pueda “leer” cuatro canales desde la primera capa, y que la normalización de entrada (medias y desviaciones típicas) se defina también para cuatro componentes. Estos cambios son los que permiten que la red aprenda a combinar información visible e infrarroja desde el inicio, en lugar de llevar a cabo la fusión mediante un preprocesado fijo.

Por último, se cuidó que los experimentos fueran fáciles de repetir. El uso de Docker ayuda a fijar el entorno, y además se mantuvo una organización de resultados que permite guardar modelos entrenados, métricas y configuraciones de cada ejecución.

# Entrenamiento y evaluación

El entrenamiento se planteó como un proceso iterativo por épocas, separando el conjunto de datos en entrenamiento y validación/test para poder medir el rendimiento al final de cada época.

En cada iteración, las imágenes fusionadas (RGB+IR) se cargan como una única entrada de 4 canales y se asocian a sus anotaciones de tipo bounding box (formato VOC). La normalización de la entrada se realizó mediante promedios y desviaciones estándar calculadas para 4 canales, de forma que la red trabaje con valores comparables entre canales desde el principio.

El flujo general fue: entrenar durante una época completa sobre el conjunto de entrenamiento, y al finalizar cada época evaluar sobre el conjunto de validación para registrar métricas de detección. Este método permite comprobar rápidamente si el modelo mejora, se estanca o empieza a degradarse, y además sirve para guardar automáticamente el mejor modelo según una métrica objetivo. En este caso, la métrica que se ha establecido como objetivo maximizar ha sido el mAP50 al priorizar la detección de personas en las imágenes. A continuación se mostrarán tanto los hiperparámetros utilizados durante el entrenamiento como las métricas que se considerarán para evaluar el modelo (aunque sea únicamente el mAP50 la que se utilice como objetivo).

## Hiperparámetros utilizados

El entrenamiento se realizó durante 10 épocas, con un tamaño de lote de 2 y una tasa de aprendizaje de 0.005. Para el cálculo de precisión y recall se usó un umbral de confianza alto (0.9), y para el cálculo de mAP se consideraron predicciones desde un umbral mínimo (0.05) para no perder información en la curva de precisión-recall. Estos valores se eligieron para tener, por un lado, una lectura exigente de calidad (precision/recall con 0.9), y por otro un mAP más representativo del comportamiento global del detector (mAP con umbral mínimo bajo).

## Métricas de evaluación

Se registraron cuatro métricas principales por época: precision, recall, mAP50 y mAP50-95. El mAP50 mide la efectividad de los bounding boxes calculados con un criterio de solapamiento relativamente permisivo, mientras que mAP50-95 es más exigente porque promedia varios umbrales de IoU y penaliza más los errores de localización. Esto es útil para diferenciar entre “detectar” y “detectar bien colocado”.

## Resultados del entrenamiento

El entrenamiento se realizó durante 10 épocas sobre LLVIP, evaluando al final de cada época con las métricas de detección descritas anteriormente.

La configuración empleada usaría hiperparámetros con los valores establecidos anteriormente, con el objetivo de obtener una evaluación exigente a nivel de falsos

positivos/negativos y, a la vez, una medida global representativa del comportamiento del detector. Se muestra a continuación la evolución a lo largo de las épocas de las métricas consideradas.

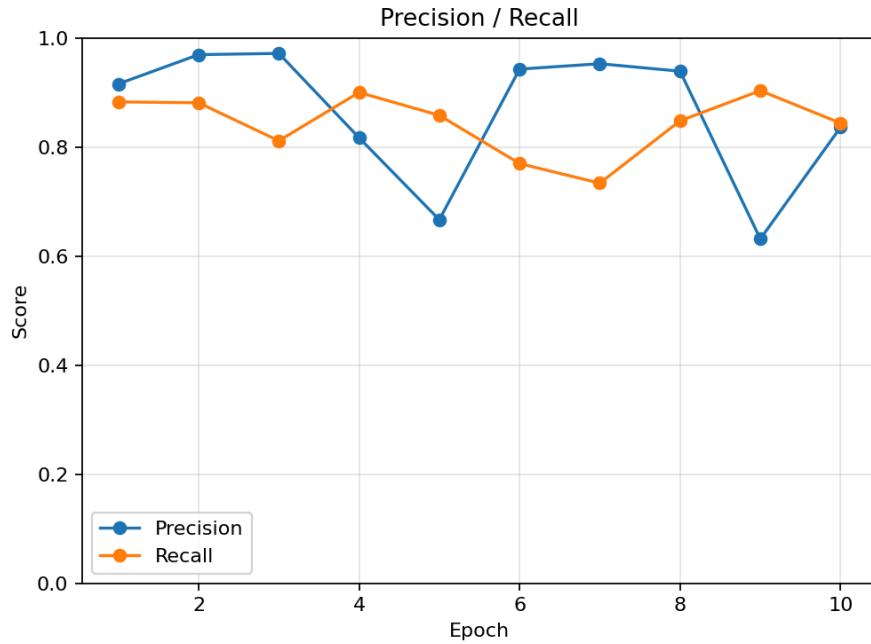


Ilustración 2. Evolución de precision y recall a lo largo de las épocas (Faster-RCNN).

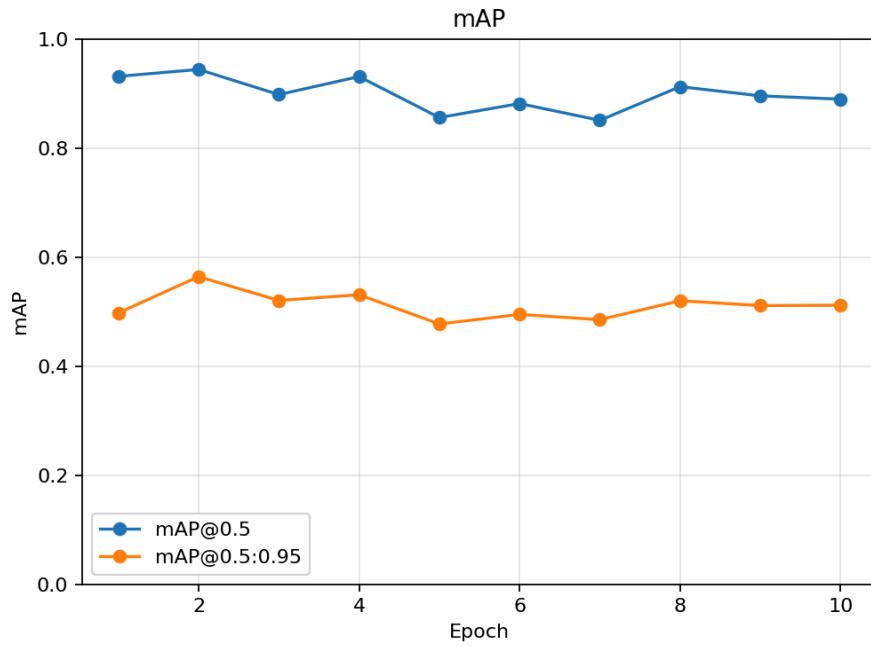


Ilustración 3. Evolución de mAP50 y mAP50-95 a lo largo de las épocas (Faster-RCNN).

En términos de rendimiento, el modelo alcanzó valores muy altos de mAP50 desde el inicio y mostró una convergencia rápida. El mejor resultado se obtuvo en la época 2.

A partir de ahí, el entrenamiento mantuvo resultados competitivos, aunque con oscilaciones moderadas entre épocas, algo habitual cuando se evalúa por épocas, y depende de cómo evoluciona el equilibrio entre localizar bien las cajas y asignarles confianza. En el conjunto completo de épocas, mAP50 se movió aproximadamente entre 0.851 y 0.944, y mAP50-95 entre 0.477 y 0.564.

Un mAP50 cercano a 0.94 indica que el modelo detecta a la mayoría de personas con un solapamiento razonable. Sin embargo, el mAP50-95 baja a valores alrededor de 0.5, lo que sugiere que, aunque detecta bien, la localización exacta (ajuste fino de la caja) es menos precisa en este problema. Esto tiene sentido en baja visibilidad: los contornos en RGB pueden ser pobres y en IR muchas veces se observan regiones más “difusas”, lo que complica ajustar con precisión los bordes de la caja, aunque la presencia de la persona sea clara.

En cuanto a precision y recall, se observan variaciones más marcadas que en mAP. La precisión osciló aproximadamente entre 0.63 y 0.97, y el recall entre 0.73 y 0.90 según la época. Estas variaciones son coherentes con haber medido precision/recall con un umbral de confianza alto (0.9): pequeños cambios en la distribución de scores del modelo pueden provocar cambios notables en cuántas detecciones superan ese umbral y, por tanto, en falsos positivos o falsos negativos. Por ejemplo, en la época 5 la precisión cae a 0.667 mientras el recall se mantiene en 0.858, y en la época 9 la precisión baja a 0.632 con recall alto (0.903). En la práctica, esto suele indicar que en esas épocas el modelo asigna confianza alta a algunas detecciones incorrectas (suben falsos positivos), aunque siga encontrando muchas personas (recall alto). Lo importante es que estas fluctuaciones no se traducen en un colapso del mAP, lo que apunta más a un ajuste fino del “calibrado” de las confianzas que a una pérdida real de capacidad de detección.

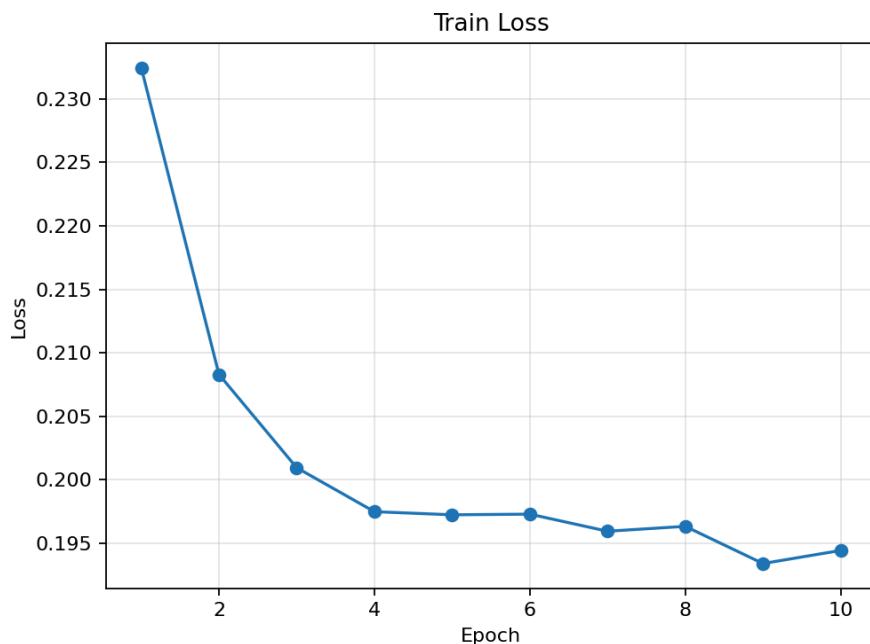


Ilustración 4. Evolución de las pérdidas a lo largo de las épocas (Faster-RCNN).

La curva de train loss desciende con fuerza al principio y después se estabiliza, pasando aproximadamente de 0.232 en la época 1 a valores cercanos a 0.194-0.196 en las últimas épocas. Esto indica una convergencia rápida: el modelo aprende pronto las características principales y luego entra en una fase de mejoras pequeñas. En detección, además, es normal que el loss y el mAP no estén perfectamente correlacionados, porque el loss mezcla varios términos (clasificación, regresión de caja, etc.) y una mejora numérica pequeña no siempre se traduce en más mAP, especialmente cuando ya se encuentra en valores altos.

Con estos resultados, tiene sentido seleccionar como “modelo final” el de la época 2 si el criterio principal es maximizar mAP50, ya que fue el que registró mejores resultados, y además, alcanzó un buen equilibrio entre precisión y recall.

Si, en cambio, se quisiera un modelo con un comportamiento más conservador o más estable según otro criterio (por ejemplo, maximizar recall o reducir falsos positivos), se podría justificar elegir otra época, aunque en este proyecto se ha priorizado la detección de personas, lo que ha llevado a establecer el mAP50 como la métrica a maximizar.

## Coste computacional y uso de GPU

En términos de coste computacional, el entrenamiento completo tardó 43.697 segundos, es decir, 728,3 minutos (aproximadamente 12,1 horas).

El tiempo medio por época fue de 4.369,6 segundos (~72,8 min), mientras que el tiempo medio por lote fue 0,494 segundos, con un percentil 95 de 0,498 segundos, lo que refleja una ejecución altamente estable sin picos grandes de latencia.

En cuanto a memoria GPU, el máximo de memoria asignada fue de 2,66 GB y la reservada de 3,72 GB, lo que implica que el entrenamiento se realizó con margen amplio respecto a la capacidad total de la Jetson AGX Orin (32 GB), dejando espacio para futuras pruebas o mejoras (como por ejemplo, aumentar resolución o ajustar batch size) sin estar limitados de forma inmediata por memoria.

# Detección y evaluación

Una vez entrenado y seleccionado el modelo, se realizó una fase de predicción y evaluación sobre el conjunto de prueba. En esta fase, cada muestra se procesa como una entrada fusionada de 4 canales (RGB+IR) y se aplica la misma normalización usada en el entrenamiento, para que las distribuciones de entrada sean consistentes y el modelo no “vea” un dominio distinto al evaluar.

La detección se planteó con dos objetivos: por un lado, generar predicciones útiles para inspección cualitativa (imágenes con cajas dibujadas) y, por otro, obtener métricas cuantitativas comparables. Para ello se trabajó con un umbral de confianza alto (0.9) cuando se habla de “detecciones finales” (las que se dibujan y las que se usan en precision/recall), y con un umbral mínimo mucho más bajo (0.05) para calcular el mAP, ya que el mAP necesita considerar también predicciones de baja confianza para construir correctamente la curva precisión/recall.

La evaluación se realizó comparando las detecciones con el *ground truth* del dataset (cajas en VOC). Se midieron precision y recall con criterio de solape  $\text{IoU} = 0.5$ , y se calculó AP a  $\text{IoU}=0.5$  (mAP50) y mAP50-95, que promedia varios niveles de IoU y es más exigente con la colocación exacta de las cajas.

## Resultados obtenidos

Con el umbral de confianza fijado en 0.9, los resultados medios fueron:

- Precision = 0.969
- Recall = 0.881
- mAP50 = 0.944
- mAP50-95 = 0.564

La lectura principal es que el modelo detecta muy bien la presencia de personas cuando se evalúa con un solape moderado (mAP50 alto) y, además, lo hace con muy pocos falsos positivos a umbral 0.9 (precision ~0.97). El recall (~0.88) indica que todavía hay un porcentaje de personas que no se detectan a ese nivel de confianza, lo cual es esperable en baja visibilidad: suelen fallar más los casos con personas a una larga distancia, parcialmente ocultas o con contraste térmico bajo respecto al fondo. La diferencia entre mAP50 y mAP50-95 vuelve a ser importante: el rendimiento cae cuando se exige mayor precisión geométrica, lo que apunta a que el detector acierta “dónde hay persona” pero el ajuste del contorno de la caja con máxima exactitud es de menor calidad en este modelo.

A continuación se muestran algunas imágenes obtenidas (detección acompañada de imágenes RGB e IR):



Ilustración 5. Imagen visible del dataset LLVIP.



Ilustración 6. Imagen infrarroja del dataset LLVIP.



Ilustración 7. Imagen con personas detectadas por el modelo entrenado con Faster-RCNN.



Ilustración 8. Imagen visible del dataset LLVIP.



Ilustración 9. Imagen infrarroja del dataset LLVIP.

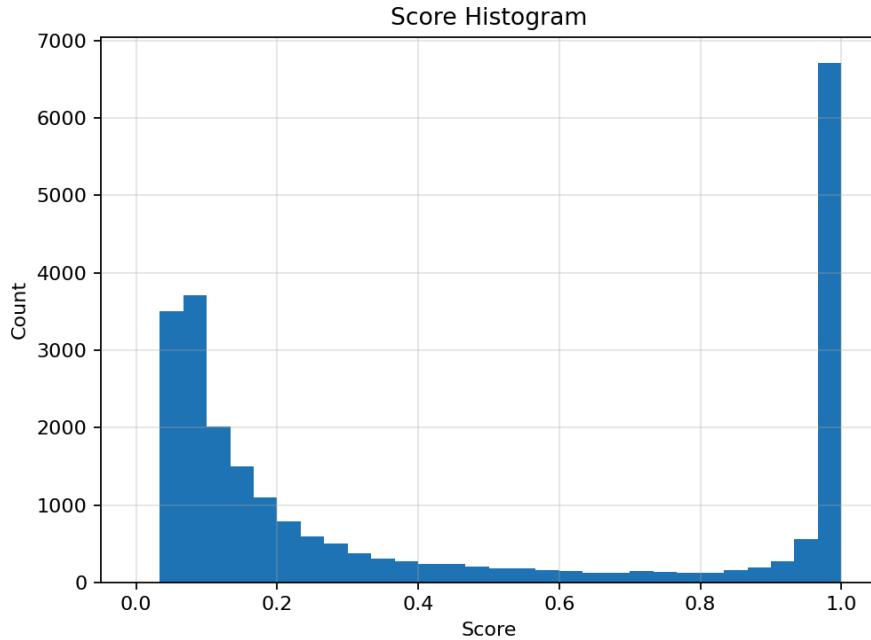


Ilustración 10. Imagen con personas detectadas por el modelo entrenado con Faster-RCNN.

## Histograma de scores y elección del umbral

El histograma de scores ayuda a justificar el umbral de 0.9. Se ve un comportamiento típico de detectores: muchas predicciones quedan con score bajo (zona 0.05-0.2) mientras que, por otro lado, aparece un bloque grande de detecciones con scores muy altos, cerca de 1.

Esto sugiere que, cuando el modelo está seguro, lo está “de verdad”, y que aplicar un umbral alto elimina la mayor parte del ruido sin perder todas las detecciones útiles.



*Ilustración 11. Histograma de scores de la predicción (Faster-RCNN).*

Esto también explica el patrón precision/recall: un umbral alto favorece precision (menos falsos positivos) a costa de perder algunas detecciones (recall algo menor). Si en alguna circunstancia interesara maximizar el recall (por ejemplo, en un sistema más “sensible”), una bajada controlada del umbral podría elevarlo, a costa probablemente de introducir más falsos positivos, por lo que en este proyecto se ha decidido mantener su valor en 0.9.

## Rendimiento computacional en predicción

Además de las métricas de detección, se registró el coste temporal y el uso de memoria durante la inferencia en GPU. En promedio, el tiempo de inferencia pura fue de 0,1166 segundos por imagen, con un FPS medio cercano a 8,57. Si se considera el “tiempo de inferencia completo por imagen” (carga, inferencia, filtrado, dibujo y guardado), el promedio asciende a 0,2364 segundos por imagen.

En la práctica, esto implica que procesar el conjunto de validación completo equivale a unos 6,7 minutos de inferencia pura y alrededor de 13,6 minutos si se incluye toda la generación de salidas (anotación y escritura a disco), tomando esos promedios como referencia.

En memoria, el pico durante la ejecución de la evaluación fue bajo: ~0,59 GB asignados y ~0,70 GB reservados, lo que confirma que la evaluación entra holgadamente en la GPU disponible.

# Comparación de entrenamiento con DETR

Para evaluar si un detector basado en *transformers* podía aportar ventajas frente al enfoque clásico, se entrenó DETR en las mismas condiciones generales que Faster R-CNN: mismo dataset LLVIP y misma lógica de evaluación por épocas usando precision, recall, mAP50 y mAP50-95. Igual que en Faster R-CNN, la entrada se mantiene como fusión temprana por apilado de canales (4 canales), de forma que la comparación mida sobre todo la arquitectura del detector y no un preprocesado distinto.

En cuanto a dinámica de aprendizaje, DETR muestra un comportamiento más “progresivo”: el train loss cae de forma bastante suave durante las 10 épocas, como se verá a continuación.

## Resultados y comparación con Faster R-CNN

En cuanto a dinámica de aprendizaje, DETR muestra un comportamiento más “progresivo”: el train loss cae de forma bastante continua durante las 10 épocas, pasando de 0.874 a 0.514, sin las estabilizaciones tempranas típicas que se ven en otros detectores.

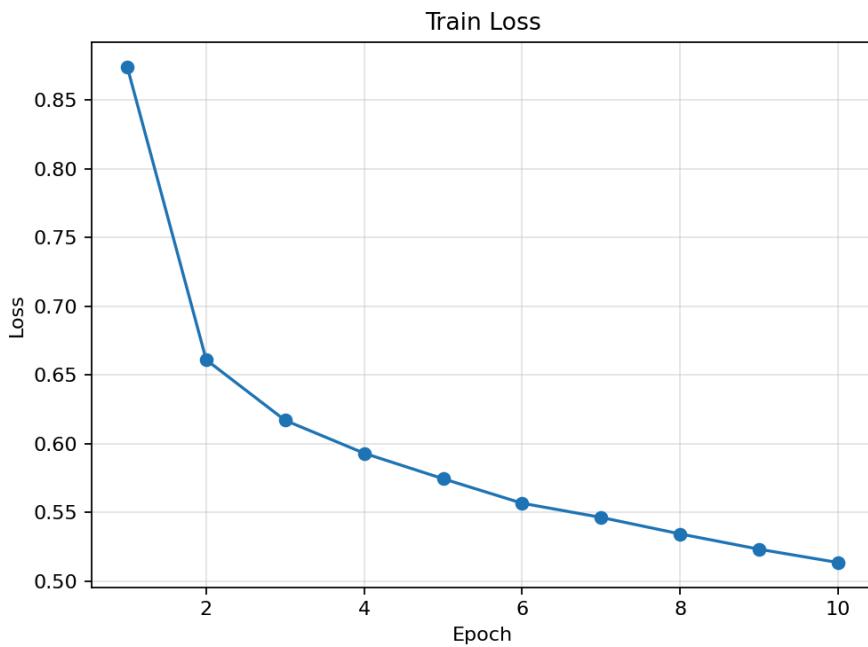


Ilustración 12. Evolución de las pérdidas a lo largo de las épocas (DETR).

Esto encaja con la idea de que DETR suele necesitar más ajuste para consolidar la correspondencia entre predicciones y objetos, y por eso es habitual que el rendimiento vaya mejorando de forma más gradual a lo largo del entrenamiento.

El mejor modelo basado en DETR se obtuvo en la época 6, y en Faster-RCNN en la época 2. La comparación deja dos ideas claras:

1. Faster R-CNN gana en mAP50 y precision. En esta tarea, esto suele traducirse en menos falsos positivos cuando se exige confianza alta, y una detección muy sólida cuando el solape requerido no es extremadamente estricto.
2. DETR mejora ligeramente mAP50-95 y recall. Que DETR suba en mAP50-95 sugiere mejor comportamiento cuando se exige mayor precisión en la colocación de los bounding boxes, y el recall más alto indica que tiende a “perder” menos personas en conjunto.

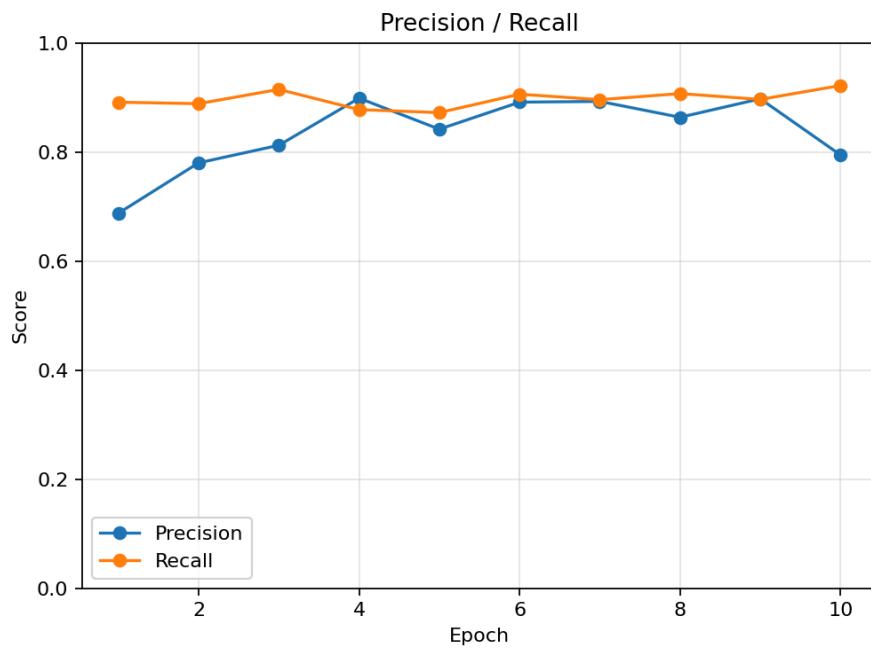


Ilustración 13. Evolución de precision y recall a lo largo de las épocas (DETR).

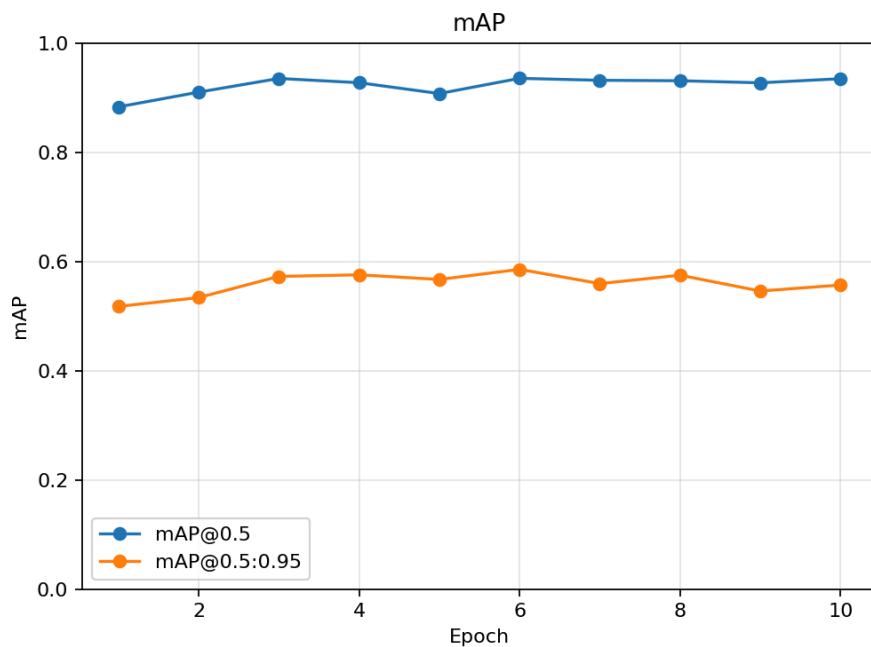


Ilustración 14. Evolución de mAP50 y mAP50-95 a lo largo de las épocas (DETR).

En las curvas por época se ve además que DETR mantiene un rendimiento bastante estable: mAP50 se mueve en torno a 0.88-0.94 y mAP50-95 en torno a 0.52-0.59, con variaciones moderadas.

En Faster R-CNN el mejor pico aparece muy pronto y luego hay más oscilación, lo que sugiere que Faster “aprende lo principal” antes pero su equilibrio de confianza/localización cambia más con el paso de las épocas.

## Coste computacional: tiempo y memoria

En tiempo total, DETR fue más costoso:

- DETR: 845,4 minutos (~14,1 horas) de entrenamiento total.
- Faster R-CNN: 728,3 minutos (~12,1 horas).

También se observa un coste por batch mayor en DETR, aunque aproximadamente igual de estable:

- DETR: 0,591 s/batch ( $p95 \approx 0,596$  s).
- Faster R-CNN: 0,494 s/batch ( $p95 \approx 0,498$  s).

En cuanto a memoria GPU consumida, DETR también sube ligeramente respecto a Faster R-CNN:

- DETR: pico ~3,26 GB asignados y ~4,14 GB reservados.
- Faster R-CNN: pico ~2,66 GB asignados y ~3,72 GB reservados.

Esto es importante porque confirma que DETR, además de ser más lento en este entorno, tiende a exigir una mayor cantidad de recursos, lo cual afecta a cuánto se puede escalar (más resolución, más épocas, etc.) manteniendo tiempos razonables.

## Conclusión de la comparación

Con estos resultados, Faster R-CNN se posiciona como mejor modelo entre ambos si el objetivo principal es maximizar tanto la precision y como el mAP50 en este problema.

DETR, si bien se queda un poco por detrás en cuanto a precision y mAP50, muestra cualidades interesantes: mejora en mAP50-95 y recall, y sostiene una evolución más estable a lo largo de las épocas de entrenamiento, lo que sugiere que puede beneficiarse de un mayor ajuste (más épocas o afinado) si se quisiera elevar su rendimiento.

| Característica / Arquitectura | Faster R-CNN                  | DETR  |
|-------------------------------|-------------------------------|---|
| Arquitectura / backbone       | Two-stage,<br>ResNet-50 + FPN | End-to-end, DETR ResNet-50<br>(transformer) |

|  |                   |                   |
|--|-------------------|-------------------|
| <b>Épocas</b>                                  | 10                | 10                |
| <b>Batch size / LR</b>                         | 2 / 0.005         | 2 / 0.005         |
| <b>Mejor época (por mAP50)</b>                 | 2                 | 6                 |
| <b>Train loss (1 → 10)</b>                     | 0.232 → 0.194     | 0.874 → 0.514     |
| <b>Tiempo total entrenamiento</b>              | ~12.1 h           | ~14.1 h           |
| <b>Tiempo medio por época</b>                  | ~72.8 min         | ~84.5 min         |
| <b>Tiempo por batch (media / p95)</b>          | 0.494 s / 0.498 s | 0.591 s / 0.596 s |
| <b>Pico memoria GPU (asignada / reservada)</b> | 2.66 GB / 3.72 GB | 3.26 GB / 4.14 GB |

Tabla 2. Comparativa de entrenamiento entre arquitectura de detector Faster-RCNN vs DETR.

# Comparación de detección con DETR

La fase de predicción con DETR se realizó sobre el mismo conjunto de prueba, la misma entrada fusionada de 4 canales (RGB+IR) y la misma normalización (medias y desviaciones calculadas para 4 canales).

El objetivo fue doble: generar detecciones para una inspección cualitativa y, a la vez, medir el rendimiento con métricas comparables a las usadas con Faster R-CNN.

En DETR, la predicción sigue una lógica ligeramente distinta a la de detectores con propuestas: el modelo genera un conjunto fijo de predicciones por imagen y después se postprocesan para obtener cajas en píxeles y scores. A partir de ahí, se filtran las predicciones de la clase *persona* y se aplica un umbral de confianza de 0.9 para considerar “detecciones finales” en precision/recall. Para el cálculo de mAP se usa un umbral mínimo más bajo (0.05) para no recortar la curva precisión–recall y poder calcular AP de forma representativa.

## Resultados en test y comparación con Faster R-CNN

Con confianza = 0.9 e IoU = 0.5, los resultados obtenidos fueron:

- Precision = 0.891
- Recall = 0.906
- mAP50 = 0.935
- mAP50-95 = 0.586

La interpretación principal es que DETR mantiene un rendimiento muy alto en detección ( $mAP50 > 0.9$ ) y destaca especialmente en dos aspectos: el recall es elevado y el mAP50-95 es competitivo, lo que indica un buen comportamiento cuando se exige mayor precisión en la localización de los bounding boxes. Usando el mismo umbral de confianza (0.9), Faster R-CNN alcanzó:

- Precision = 0.969
- Recall = 0.881
- mAP50 = 0.944
- mAP50-95 = 0.564

La comparación deja un patrón claro: Faster R-CNN es más conservador y preciso con umbral alto, su precision es mayor, lo que sugiere menos falsos positivos cuando solo se aceptan detecciones de alta confianza; DETR recupera más casos difíciles, mejora el recall y también sube mAP50-95, lo que apunta a detecciones útiles en más situaciones y, en promedio, mejor ajuste cuando se evalúa con IoU más exigentes.

Dicho de forma simple: si la prioridad es “no equivocarse” cuando se muestra una detección (minimizar falsos positivos), Faster R-CNN sale mejor parado; por otro lado, si la prioridad es “no perder personas” y mantener buen rendimiento cuando se exige localización más fina, DETR muestra una ligera ventaja.

A continuación se muestran algunas imágenes obtenidas (mismas imágenes RGB-IR que antes):



Ilustración 15. Imagen con personas detectadas por el modelo entrenado con DETR.



Ilustración 16. Imagen con personas detectadas por el modelo entrenado con DETR.

## Histograma de scores

El histograma de *scores* en DETR puede resultar “extraño” a primera vista, al tener un pico enorme cerca de 0 y un grupo pequeño cerca de 1. Esto no es necesariamente malo, de hecho, es bastante interesante: en DETR la mayoría de predicciones por imagen terminan representando fondo/no-objeto, porque el modelo siempre genera un número fijo de consultas (queries) y solo unas pocas acaban asignadas a objetos reales; el resto se entrena explícitamente como “no-object” y por eso su probabilidad de ser un objeto cae casi a cero. Las detecciones reales, en cambio, se concentran en *scores* altos. Por tanto, este histograma es coherente con el funcionamiento del modelo y apoya el uso de un umbral alto para quedarnos con detecciones fiables.

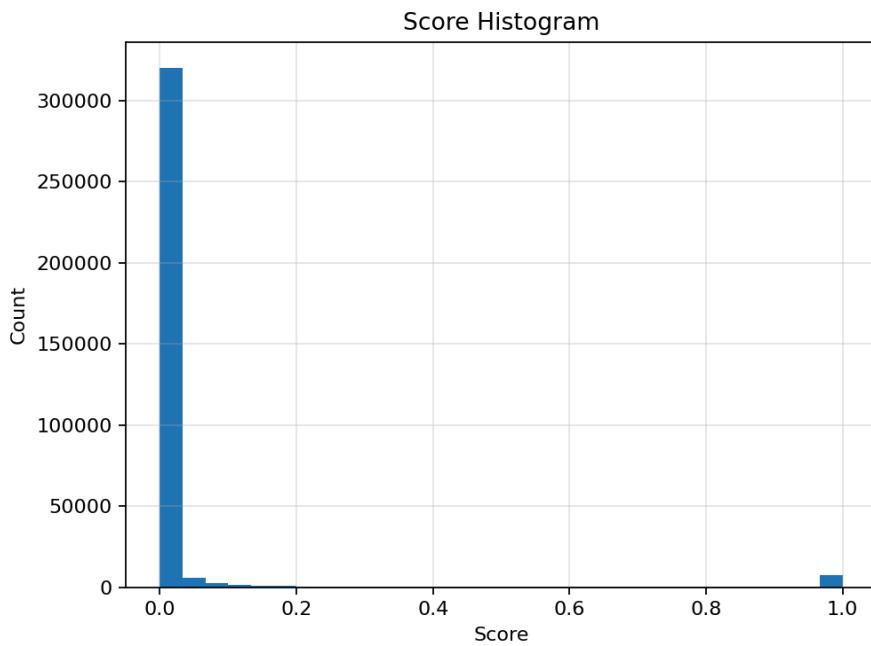


Ilustración 17. Histograma de scores de la predicción (DETR).

## Rendimiento computacional en predicción

En cuanto a inferencia, DETR mostró:

- Tiempo de inferencia medio: 0.1118 s/imagen → ~8.94 FPS
- Tiempo total por imagen (carga + inferencia + postproceso/guardado): 0.2374 s/imagen

Frente a Faster R-CNN, los tiempos son muy parecidos, con una ligera ventaja en inferencia pura para DETR (más FPS), mientras que el tiempo total por imagen queda prácticamente igual al incluir lectura y escritura, lo que supone una mayor carga. En memoria GPU, el pico fue bajo ( $\approx$  0.56 GB asignados y 0.65 GB reservados), de nuevo sin limitaciones por capacidad.

## Conclusión de la comparación

| Característica / Arquitectura                               | Faster R-CNN                  | DETR   |
|---|-------------------------------|--|
| Precision / Recall (conf=0.9)                               | 0.969 / 0.881                 | 0.891 / 0.906  |
| mAP50   | 0.944                         | 0.935  |
| mAP50-95  | 0.564                         | 0.586  |
| Inferencia (solo modelo)                                    | 0.1166 s/img                  | 0.1118 s/img   |
| FPS (solo modelo)   | 8.57                          | 8.94   |
| Tiempo total por imagen<br>(incluye E/S y guardado)         | 0.2364 s/img                  | 0.2374 s/img   |
| Pico memoria GPU en<br>inferencia (asignada /<br>reservada) | 0.59 GB / 0.70<br>GB          | 0.56 GB / 0.65 GB  |
| Histograma de scores  | Más “normal”<br>tras filtrado | Pico enorme en 0 (mucho<br><i>no-objeto</i> ) + pocas detecciones muy<br>altas |

Tabla 3. Comparativa de métricas de detección entre modelos con arquitectura de detector Faster-RCNN vs DETR.

# Conclusiones y trabajo futuro

En este trabajo se ha logrado establecer y validar la detección multimodal de personas en diferentes entornos con características adversas. Mediante los trabajos de investigación y la implementación realizada, se ha conseguido desarrollar con éxito un algoritmo de apilado de canales que transforma pares de imágenes independientes en tensores unificados de 4 canales. Esta implementación permite aprovechar arquitecturas de *Deep Learning* preexistentes, como las de detectores DETR o Faster R-CNN, para procesar información térmica sin necesidad de diseñar redes complejas desde cero, validando así la estrategia de *Early Fusion* como una solución viable y de bajo coste computacional para el proyecto.

Asimismo, la elección de dataset LLVIP ha conseguido una gran optimización del sistema, esta elección ha resultado determinante para la calidad de los datos de entrada. A diferencia de otros datasets analizados durante la fase de investigación, como KAIST o FLIR ,LLVIP ha eliminado la necesidad de aplicar algoritmos de emparejamiento geométrico adicionales. Esta característica ha garantizado una correspondencia píxel a píxel perfecta entre la información visual y la térmica.

Finalmente, el análisis de los datos y las pruebas confirman la robustez del sistema en condiciones adversas, demostrando que la integración del espectro infrarrojo compensa las limitaciones de los sensores RGB convencionales. Se ha verificado que la firma térmica humana proporciona características excluyentes y robustas incluso en condiciones de oscuridad total o baja iluminación, cumpliendo con el objetivo principal de mejorar la fiabilidad del sistema de percepción en entornos de baja visibilidad donde la cámara visible por sí sola resultaría insuficiente.

A partir de todo lo desarrollado en este proyecto, se proponen algunas mejoras orientadas a evolucionar el sistema hacia un proceso real. En primer lugar, se identifica la validación en hardware (*Edge Computing*), concretamente la portabilidad del modelo a plataformas de bajo consumo como la NVIDIA Jetson. Este proceso implica la aplicación de técnicas de cuantización de la red neuronal (a formatos FP16 o INT8) y su optimización mediante librerías como TensorRT, con el objetivo de garantizar una inferencia en tiempo real compatible con la navegación robótica autónoma.

Dado que el dataset actual se centra en condiciones de baja iluminación, se propone robustecer el entrenamiento mediante el uso de técnicas avanzadas de *Data Augmentation* o redes generativas antagónicas (GANs) permitiría sintetizar perturbaciones como niebla y humo en las imágenes, con el objetivo de facilitar situaciones de emergencias.

Por último, la validación de este sistema permite su aplicación en una amplia variedad de entornos. En primer lugar, la técnica es aplicable al seguimiento y al rescate de personas atrapadas en incendios. En el ámbito del transporte, cabe resaltar la detección de peatones en condiciones difíciles en vehículos autónomos, la detección de personas u objetos en la zona de pista de aviación con humo o niebla y en el ámbito ferrocarril, en túneles de baja iluminación, detección o seguimiento con los problemas presentados.

En el ámbito industrial, se implantaría para la supervisión de trabajadores y un control en la seguridad de los operarios y en el ámbito civil, se podría llegar a usar en eventos masivos

como conciertos, en los que garantizar la seguridad pública es algo primordial para asegurar que no ocurran situaciones de pánico y además se podría utilizar en la gestión de accidentes urbanos, implementandolo en los sistemas de vigilancia tradicionales.

# Referencias

- [1] S. R. Schnelle and A. L. Chan, "Enhanced target tracking through infrared-visible image fusion", 14th International Conference on Information Fusion, Chicago, IL, USA, 2011, pp. 1-8.
- [2] Wang, H., An, W., Li, L., Li, C., Zhou, D.: Infrared and visible image fusion based on multi-channel convolutional neural network. *IET Image Process.* 16, 1575–1584 (2022).
- [3] Liu, J., Zhang, S., Wang, S., & Metaxas, D. N. (2016). Multispectral deep neural networks for pedestrian detection. arXiv preprint arXiv:1611.02644.
- [4] Jia, Xinyu, et al. "LLVIP: A Visible–Infrared Paired Dataset for Low-Light Vision." *LLVIP Dataset*, 2021, <https://bupt-ai-cz.github.io/LLVIP/>.
- [5] Hwang, S. *Multispectral Pedestrian Detection (KAIST Dataset)*. RGB-T Pedestrian Detection, 2015. <https://soonminhwang.github.io/rGBT-ped-detection/>
- [6] FLIR Systems, Inc. *FLIR ADAS Dataset*. FLIR Automotive Solutions, 2018. <https://oem.flir.com/es-ES/solutions/automotive/adas-dataset-form/>