UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Grado en Ciencia de Datos e Inteligencia Artificial

Trabajo Fin de Grado

Identificación de la planta de edificios mediante segmentación semántica a partir de imágenes del PNOA

Autor: Jorge Pastor Velasco

Tutor: Francisco Serradilla García

Codirector: Miguel Ángel Manso Callejo

Madrid, 2 de julio de 2025

Este Trabajo de Fin de Grado ha sido depositado en la ETSI de Sistemas Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ciencia de Datos e Inteligencia Artificial

Título: Identificación de la planta de edificios mediante segmentación semántica a partir de imágenes del PNOA

Julio de 2025

Autor: Jorge Pastor Velasco
Tutor: Francisco Serradilla García
Codirector: Miguel Ángel Manso Callejo

Departamento de Sistemas Informáticos
ETSI de Sistemas Informáticos
Universidad Politécnica de Madrid

Agradecimientos

A mi madre y a mi hermana, por su ayuda y su apoyo incondicional a lo largo de todo este camino.

A mis amigos, por saber sacarme una sonrisa y animarme cuando más lo necesitaba.

A mis compañeros, porque sin su apoyo y sin esas tardes de gimnasio que servían para desconectar, no sé si esto habría sido posible.

A Miguel Ángel Manso y Francisco Serradilla, por estar siempre disponibles cuando he necesitado su orientación y consejo.

Por último, quiero agradecer a la Cátedra IGN y CNIG sobre Tecnologías Aplicadas a la Información Geoespacial, por financiar este proyecto y facilitarme todos los recursos y datos necesarios para llevarlo a cabo.

Resumen

Este trabajo presenta el desarrollo y evaluación de distintos sistemas basados en segmentación semántica para la identificación automática de la planta de edificios a partir de ortofotografías aéreas del Plan Nacional de Ortofotografía Aérea (PNOA). El proyecto se enmarca en el uso de técnicas de aprendizaje profundo (DL), particularmente redes neuronales convolucionales (CNNs), aplicadas a imágenes RGB de alta resolución.

Para abordar el problema, se ha realizado un estudio del estado del arte en segmentación semántica binaria, incluyendo una revisión de arquitecturas ampliamente utilizadas como DeepLabV3+, PAN o UPerNet, todas ellas accesibles a través de la librería `segmentation_models.pytorch`. Inicialmente, se ha trabajado con el dataset INRIA para entrenar y comparar diferentes modelos, utilizando métricas como IoU, F1-score, Dice y BCE. A partir de los modelos con mejor desempeño, se ha aplicado transfer learning sobre un conjunto de datos personalizado, creado a partir de imágenes del PNOA proporcionadas por el Instituto Geográfico Nacional (IGN), cuyas máscaras han sido revisadas y validadas manualmente.

Los modelos seleccionados han sido evaluados sobre regiones geográficas de España, para analizar la capacidad de los modelos en condiciones reales y valorar su utilidad para aplicaciones como cartografía, planificación urbana o análisis del territorio.

Palabras clave: segmentación semántica, CNN, imágenes aéreas, PNOA, identificación de edificios, aprendizaje profundo, transfer learning.

Abstract

This work presents the development and evaluation of various systems based on semantic segmentation for the automatic identification of building footprints from aerial orthophotos of the Plan Nacional de Ortofotografía Aérea (PNOA). The project is framed within the use of deep learning (DL) techniques, particularly convolutional neural networks (CNNs), applied to high-resolution RGB imagery.

To address the problem, a study of the state of the art in binary semantic segmentation has been conducted, including a review of widely used architectures such as DeepLabV3+, PAN and UPerNet, all available through the `segmentation_models.pytorch` library. Initially, the INRIA dataset has been used to train and compare different models, employing metrics such as IoU, F1-score, Dice, and BCE. Based on the best-performing models, transfer learning has been applied to a custom dataset created from PNOA images provided by the Instituto Geográfico Nacional (IGN), whose masks have been manually reviewed and validated.

The selected models have been evaluated over geographical regions of Spain to analyze their performance under real-world conditions and to assess their applicability in fields such as cartography, urban planning, or territorial analysis.

Keywords: semantic segmentation, CNN, aerial imagery, PNOA, building footprint identification, deep learning, transfer learning.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En los últimos años, el uso de imágenes aéreas de alta resolución ha cobrado gran importancia en campos como la cartografía, la planificación urbana, la monitorización territorial o la gestión del suelo. Una de las tareas fundamentales en estos contextos es la identificación precisa de la planta de edificios, ya que, permite inferir la ocupación del espacio construido y realizar análisis espaciales relevantes para la toma de decisiones.

Tradicionalmente, esta identificación se ha llevado a cabo de forma manual, lo cual implica una gran inversión de tiempo y recursos humanos, especialmente cuando se trata de áreas geográficas extensas. En respuesta a esta problemática, el presente proyecto propone el uso de técnicas de aprendizaje profundo (DL), concretamente modelos de segmentación semántica basados en redes neuronales convolucionales (CNN), para automatizar la detección de edificios en ortofotografías aéreas del Plan Nacional de Ortofotografía Aérea (PNOA), proporcionadas por el Instituto Geográfico Nacional (IGN).

A través del entrenamiento de diversos modelos de segmentación sobre el dataset INRIA y su posterior adaptación mediante *transfer learning* a un conjunto de datos personalizado basado en imágenes del PNOA, se busca evaluar el rendimiento de estas arquitecturas en un entorno real. El resultado final es un conjunto de modelos seleccionados capaces de identificar de forma automática y precisa las plantas de edificios a partir de imágenes RGB aéreas, con potencial para integrarse en flujos de trabajo de análisis territorial automatizado.

Este proyecto no solo aborda una problemática de alta relevancia práctica, sino que también explora el uso de tecnologías punteras en visión por computador, aportando una solución escalable y reutilizable.

1.1 Objetivos

El objetivo general del proyecto es desarrollar y comparar el desempeño de diferentes modelos basados en segmentación semántica que permita la identificación automática de edificios en imágenes aéreas RGB, optimizando así los procesos de análisis espacial en aplicaciones como cartografía, urbanismo y gestión territorial.

Objetivos específicos

- Realizar un estudio del estado del arte en segmentación semántica aplicada a imágenes aéreas, incluyendo las arquitecturas más destacadas.
- Entrenar distintos modelos (DeepLabV3+, PAN, UPerNet) sobre el dataset INRIA para establecer una comparativa de rendimiento.
- Crear un conjunto de datos personalizado a partir de imágenes del PNOA, con máscaras validadas manualmente para su uso en entrenamiento y evaluación.
- Aplicar *transfer learning* desde los modelos previamente entrenados sobre INRIA al nuevo conjunto del PNOA.
- Evaluar el rendimiento de los modelos sobre regiones geográficas reales mediante métricas como IoU, F1-score, Dice y BCE.
- Analizar los resultados obtenidos y discutir la aplicabilidad práctica de los modelos desarrollados.
- Estudiar el impacto social y medioambiental de la automatización en este tipo de tareas y proponer posibles líneas futuras de trabajo.

1.2 Estructura de la memoria

La estructura del presente trabajo se organiza del siguiente modo:

- **Resumen:** Presenta una visión general del trabajo realizado, incluyendo los objetivos y metodología.
- **Introducción:** Se contextualiza el problema a resolver, se describe la motivación del proyecto y se detallan los objetivos y la estructura de la memoria.
- **Estado del arte:** Se revisan las tecnologías más relevantes en el ámbito de la segmentación semántica, prestando especial atención a las arquitecturas aplicables al análisis de imágenes aéreas.
- **Desarrollo:** Se describe el conjunto de datos utilizado, el preprocesado realizado, los modelos empleados y las técnicas de entrenamiento, así como la adaptación de las arquitecturas seleccionadas a imágenes del PNOA.
- **Resultados:** Se presentan y analizan los resultados obtenidos, comparando el rendimiento de los distintos modelos sobre los conjuntos de datos utilizados.
- **Conclusiones:** Se resumen los logros alcanzados y se discute el impacto del trabajo, incluyendo una reflexión sobre posibles mejoras, líneas futuras y el impacto social y medioambiental de la automatización de estas tareas.
- **Bibliografía:** Se recopilan todas las referencias utilizadas a lo largo del desarrollo del trabajo.

Capítulo 2

Estado del arte

2.1 Segmentación semántica y visión por computador

La visión por computador es un campo de la inteligencia artificial enfocado en la interpretación automática de imágenes y vídeo [44]. En este ámbito, las redes neuronales convolucionales (CNN) se han convertido en la metodología predominante, superando métodos tradicionales en tareas como la clasificación de imágenes, la detección de objetos o la segmentación [44]. Estas redes, inspiradas en la corteza visual animal, aprenden jerarquías espaciales de características mediante capas de convolución, pooling y conexiones totalmente conectadas [44]. La segmentación semántica, en particular, consiste en asignar una etiqueta de clase a cada píxel de la imagen, clasificando cada píxel en una clase semántica [18]. En este apartado veremos en qué consisten las redes neuronales convolucionales, qué es la segmentación semántica, el rol del backbone y su importancia dentro de las arquitecturas de segmentación semántica, qué es el transfer learning, y aplicaciones reales y uso de modelos de segmentación semántica.

2.1.1. Redes Neuronales Convolucionales y Mapas de Características

La arquitectura básica de una CNN incluye capas de convolución que aplican filtros (kernels) deslizantes sobre la imagen de entrada. Cada filtro produce un mapa de características (feature map) que resalta patrones locales (por ejemplo, bordes horizontales o verticales) en la imagen [44]. Al aplicar múltiples filtros en paralelo se generan varios mapas de características, cada uno de los cuales representa distintos atributos locales de la imagen [44]. Tras las capas de convolución (y de pooling, que reduce la resolución espacial), las CNN tradicionales incluyen una o varias capas totalmente conectadas que combinan las características extraídas para producir una predicción final de clase global [44]. En la clasificación a nivel global, toda la imagen se etiqueta con una sola clase (por ejemplo, una CNN entrenada en ImageNet podría etiquetar una imagen como “gato”).

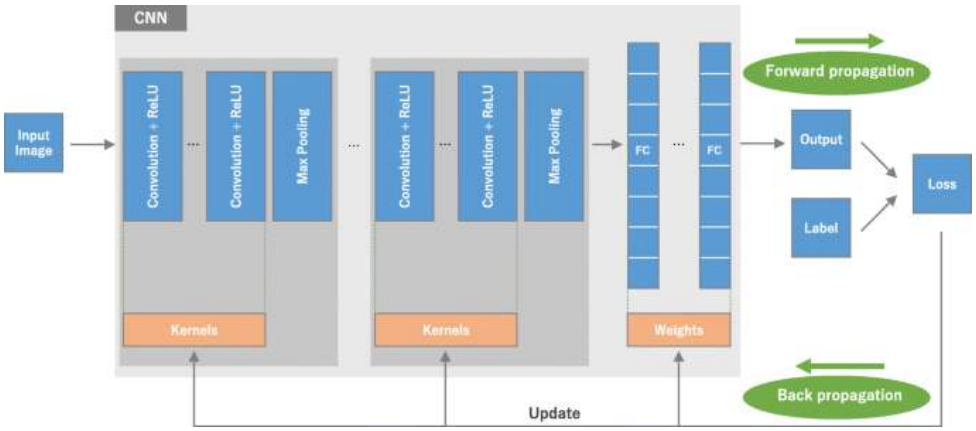


Figura 2.1: Esquema general de una red neuronal convolucional (CNN). La arquitectura incluye capas de convolución, agrupamiento (pooling) y totalmente conectadas. [44].

Por el contrario, la clasificación a nivel de píxel (como en la segmentación semántica) requiere que la red genere una predicción por cada píxel de la entrada. Una forma común de lograrlo es convertir una CNN de clasificación en una red totalmente convolucional (FCN) eliminando las capas densas y empleando convoluciones para producir un mapa de salida de la misma resolución que la imagen de entrada. Se ha demostrado que una FCN entrenada punto a punto (píxel a píxel) puede generar eficientemente predicciones densas para tareas de segmentación [23]. En este enfoque, los mapas de características de capas profundas aportan información semántica global, mientras que los de capas superficiales retienen detalles espaciales finos, permitiendo así segmentaciones precisas [23, 18].

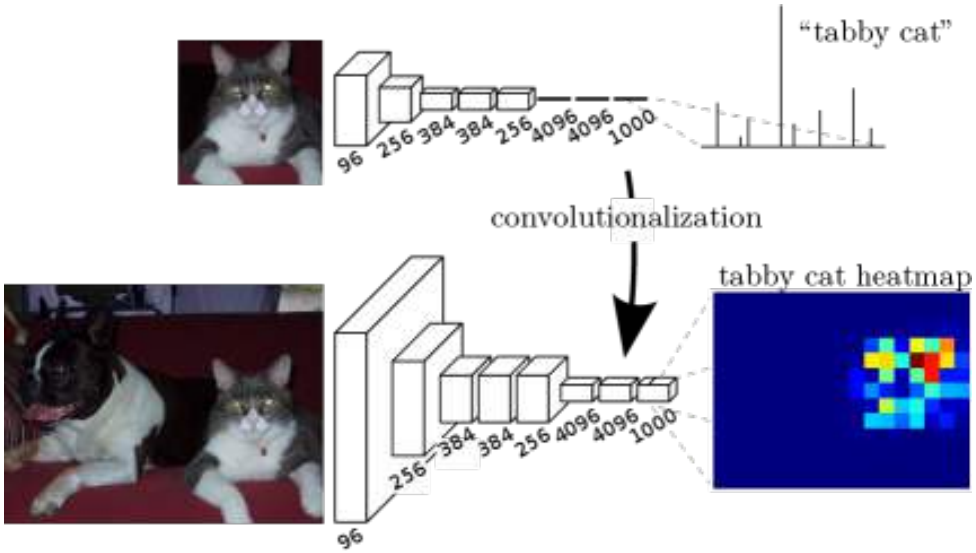


Figura 2.2: Transformación de una red convolucional clásica en una red totalmente convolucional (FCN) para segmentación semántica. La salida es un mapa denso de clases a nivel de píxel. [23].

2.1.2. Segmentación semántica

La segmentación de imágenes se puede dividir en tres subcategorías principales: semántica, por instancia y panóptica [13]. La segmentación semántica implica asignar a cada píxel una etiqueta de clase que representa su categoría semántica [38, 13]. Es decir, todos los

píxeles que corresponden a la misma clase (por ejemplo, personas) reciben la misma etiqueta, distinguiéndolos de regiones de fondo u otras clases. En contraste, la segmentación por instancia detecta cada objeto individual en la imagen y genera una máscara de segmentación con un identificador único para cada instancia [13]. La segmentación panóptica combina ambos enfoques: asigna a cada píxel tanto una etiqueta semántica como un identificador de instancia cuando corresponde [13, 17].

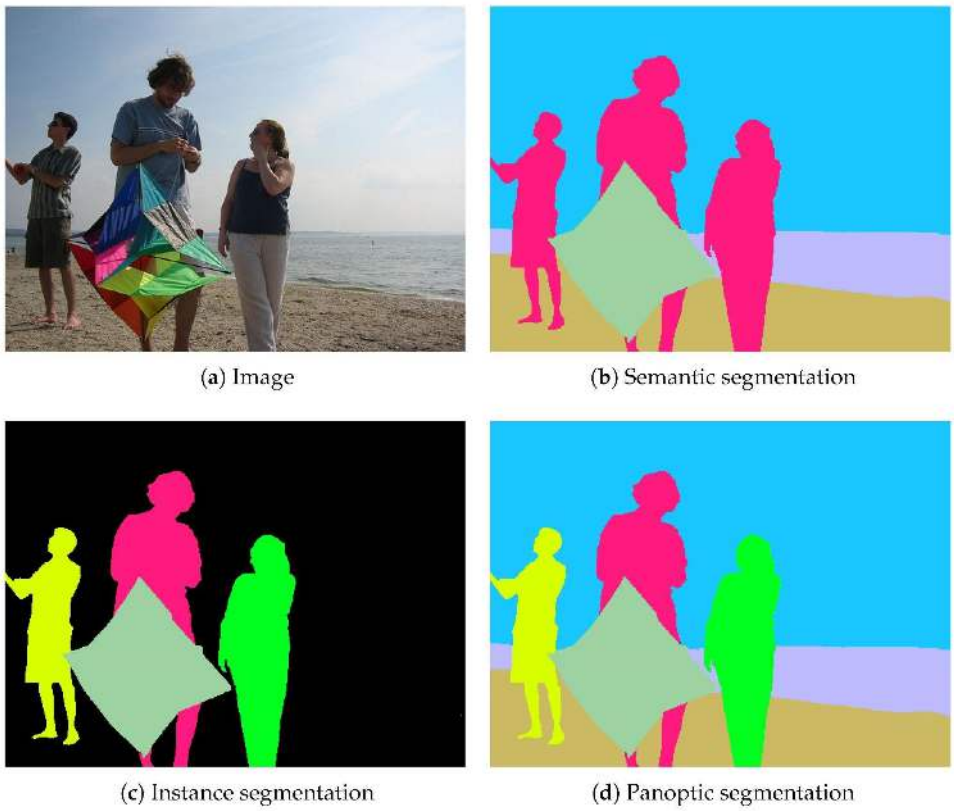


Figura 2.3: Ejemplo de los tipos de segmentación.

La implementación moderna de la segmentación semántica se basa en redes neuronales profundas. Las Fully Convolutional Networks (FCN) pueden procesar imágenes de entrada de tamaño arbitrario y producir una predicción densa de etiquetas píxel a píxel [23]. Se ha demostrado que una FCN entrenada de extremo a extremo supera el estado del arte en tareas de segmentación semántica [23]. Para recuperar la información espacial perdida por las capas de pooling, muchas arquitecturas adoptan un diseño *encoder-decoder*: el encoder extrae características reduciendo la resolución espacial, mientras el decoder aplica operaciones de upsampling para reconstruir un mapa de segmentación de la misma resolución que la imagen original [34]. Un ejemplo destacado es la U-Net [34]. El resultado es un mapa de segmentación donde cada píxel coloreado refleja la clase semántica asignada [13].

Estos modelos se entrenan de forma supervisada con grandes conjuntos de datos anotados píxel a píxel (por ejemplo, PASCAL VOC o Cityscapes [7]) y se evalúan mediante métricas como la intersección sobre la unión (IoU). La segmentación semántica es esencial para la comprensión detallada de escenas. Al generar mapas de segmentación completos, permite distinguir objetos y regiones contextuales con alta precisón [38]. A diferencia de

un clasificador de imágenes (que solo identifica clases presentes) o un detector de objetos (que localiza entidades con cajas delimitadoras), la segmentación semántica delimita las formas reales de los objetos a nivel de píxel [13]. Esto resulta crucial en aplicaciones como la conducción autónoma, donde se segmentan carriles, vehículos y peatones en cada fotograma [38], así como en el diagnóstico médico (segmentación de tejidos u órganos en imágenes médicas), donde la distinción precisa de regiones es vital.

2.1.3. El rol del backbone en la segmentación

El término backbone (o "columna vertebral") en segmentación semántica se refiere a la parte de la red convolucional encargada de extraer las características básicas de la imagen. Generalmente, el backbone es una CNN profunda originalmente diseñada para clasificación (por ejemplo VGG, ResNet, DenseNet) que se adapta al problema de segmentación [10]. Esta red procesa la imagen de entrada mediante capas jerárquicas: las capas iniciales capturan rasgos locales (bordes, texturas) y las capas profundas obtienen características más abstractas (formas u objetos completos) [10]. Habitualmente, el backbone se inicializa con pesos pre-entrenados en grandes conjuntos de imágenes (como ImageNet), lo que acelera el entrenamiento y mejora la calidad de las características extraídas para la segmentación [10]. En este sentido, el backbone actúa como el encoder de la arquitectura de segmentación, generando mapas de características a resoluciones cada vez menores pero ricos en información semántica.

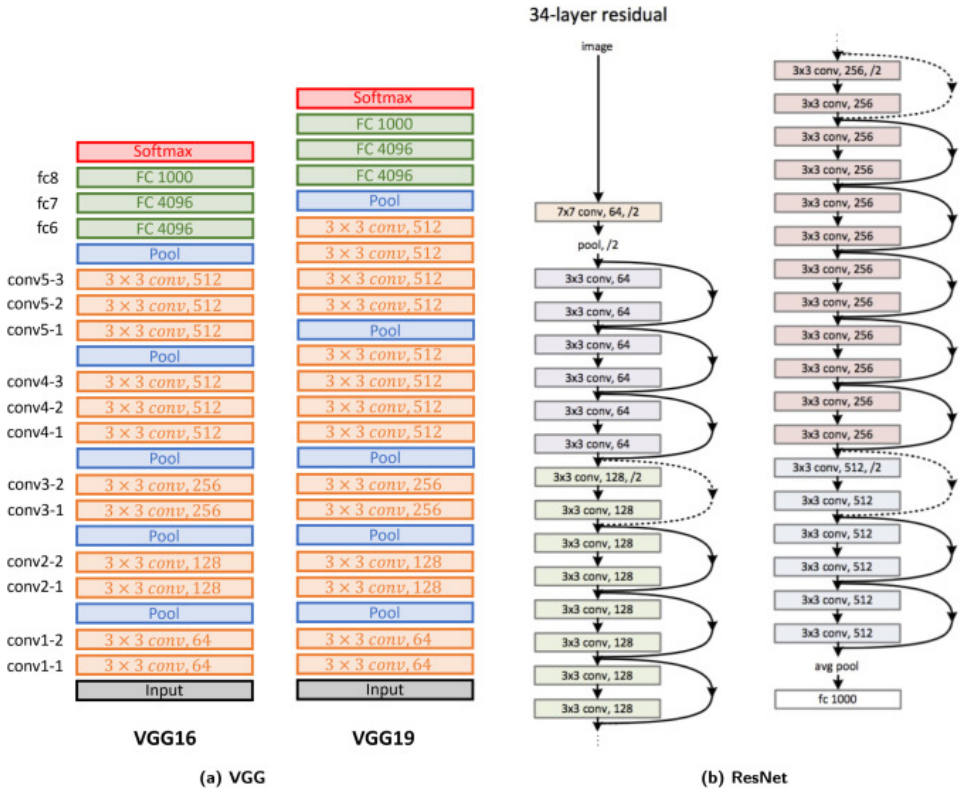


Figura 2.4: Arquitecturas de backbones CNN (VGG vs. ResNet).

En la mayoría de las arquitecturas, el backbone alimenta directamente el resto del modelo. Por ejemplo, en un esquema clásico de encoder-decoder, el encoder es justamente el backbone, mientras que el decoder se encarga de reconstruir la predicción pixel a pixel.

Los backbones más comunes en segmentación son versiones adaptadas de redes de clasificación populares (por ejemplo, convertir las capas totalmente conectadas en convoluciones en VGG o ResNet) [10]. En U-Net, por ejemplo, el encoder es un backbone como ResNet, y sus mapas de características se concatenan con el decoder correspondiente mediante skip connections, lo que permite conservar detalles espaciales finos durante la decodificación. Estas conexiones de salto (skip) enlazan capas intermedias del encoder con el decoder, aportando información de alta resolución perdida en el proceso de downsampling. De esta forma, el backbone provee un conjunto jerárquico de características (multiescala) que el decoder utilizará para generar la máscara final de segmentación.

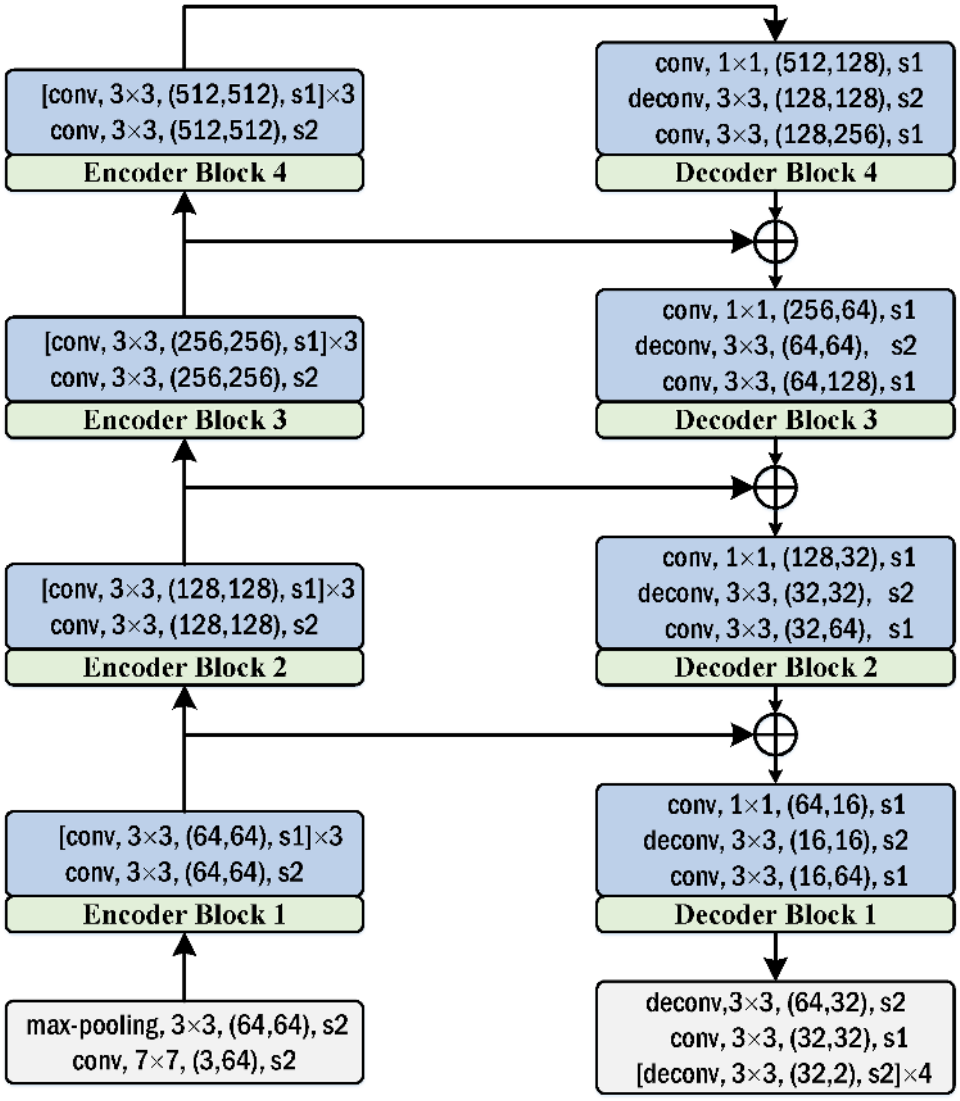


Figura 2.5: Arquitectura encoder-decoder para segmentación semántica usando un backbone ResNet-18 en el encoder.

El backbone también puede combinarse con módulos contextuales para formar el modelo completo. Por ejemplo, DeepLabV3+ extiende la familia DeepLab incluyendo un pequeño decodificador y módulos de convolución dilatada (Atrous Spatial Pyramid Pooling) para enriquecer el contexto multiescala [2]. En DeepLabV3+ se usa un backbone como Xception o MobileNet como encoder; a sus salidas se aplican convoluciones con distinto dilations (ASPP) y luego un decoder que refina los bordes de los objetos [2]. De

manera análoga, UPerNet es un esquema de segmentación que toma cualquier backbone visual (ResNet, ConvNeXt, Swin Transformer, etc.) y le aplica un Feature Pyramid Network (FPN) junto con un módulo de Pyramid Pooling, capturando características a múltiples escalas antes de predecir las etiquetas. En ambos casos el backbone sigue siendo la "columna vertebral" que provee las características base; los módulos adicionales (decodificador, FPN, pooling piramidal, etc.) se integran sobre esas características para producir la segmentación final.

La elección del backbone influye directamente en el rendimiento del modelo. Un backbone más profundo o especializado extraerá características más ricas, pero incrementa la complejidad computacional. Por ello se exploran versiones ligeras (MobileNet, EfficientNet) o variantes con convoluciones eficientes (separable, dilated, etc.). Estudios recientes han mostrado que incluso arquitecturas de segmentación relativamente simples (encoder-decoder) con un backbone tipo ResNet modificado pueden igualar el desempeño de diseños mucho más complejos (por ejemplo, HRNet) en benchmarks de segmentación [25]. En resumen, el backbone en segmentación es la pieza fundamental para la extracción de características visuales; su capacidad de codificar información semántica será aprovechada y refinada por los módulos de decodificación, skip-connections y cabezales de segmentación para generar las máscaras finales [10].

2.1.4. Transfer learning en segmentación semántica

El aprendizaje por transferencia es una técnica mediante la cual se aprovecha el conocimiento adquirido en una tarea (o dominio) fuente para mejorar el desempeño en una tarea objetivo relacionada [48]. En redes profundas, esto suele implicar reutilizar las capas iniciales de una CNN previamente entrenada (por ejemplo, en un conjunto de clasificación grande como ImageNet) como extractor de características, y adaptar (fine-tuning) el resto de la red al nuevo dominio. De este modo, se reduce la necesidad de entrenar un modelo complejo desde cero y con enormes volúmenes de datos [48]. Sin embargo, esta transferencia no siempre mejora el aprendizaje: si los dominios fuente y objetivo son muy desiguales, el conocimiento transferido puede resultar poco útil o incluso perjudicial (fenómeno conocido como transferencia negativa) [48]. En la práctica se busca que las tareas compartan patrones generales (por ejemplo, bordes y texturas en imágenes), de modo que las capas convolucionales aprendidas en la tarea fuente extraigan características igualmente útiles para la nueva tarea [48].

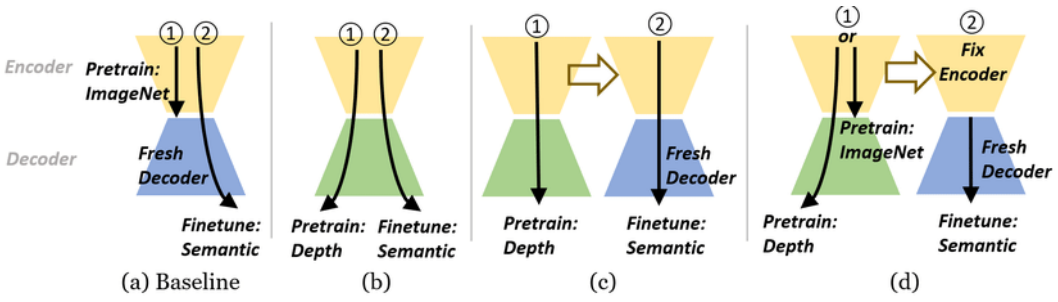


Figura 2.6: Esquemas comunes de transferencia de aprendizaje en segmentación semántica: (a) encoder preentrenado en clasificación, (b) encoder preentrenado con estimación de profundidad, y (d) encoder preentrenado fijo durante entrenamiento.

Las ventajas del aprendizaje por transferencia en aprendizaje profundo son especialmente evidentes cuando los datos etiquetados son escasos o costosos de obtener. Entre sus beneficios se incluyen:

- **Reducción de datos anotados requeridos:** al aprovechar modelos preentrenados, se necesita menos información etiquetada en la tarea objetivo para alcanzar un buen rendimiento [48].
- **Aceleración del entrenamiento:** iniciar el entrenamiento con pesos preentrenados acelera la convergencia; por ejemplo, estudios muestran que las redes inicializadas con pesos de ImageNet convergen en aproximadamente un 20 % menos iteraciones que las inicializadas aleatoriamente. Además, suelen alcanzar mayor precisión inicial en épocas tempranas.
- **Mejor generalización:** partir de parámetros ya entrenados actúa como regularizador, reduciendo el sobreajuste en conjuntos pequeños. De hecho, las CNN entrenadas en ImageNet se usan rutinariamente como inicialización estándar para nuevas tareas [14].

En el contexto de la segmentación semántica —donde se asigna una etiqueta a cada píxel de la imagen— el transfer learning se aplica típicamente mediante arquitecturas encoder–decoder (p. ej. U-Net). En estas redes, la parte encoder suele construirse con una CNN preentrenada en clasificación (por ejemplo, VGG o ResNet entrenadas en ImageNet) y se conserva gran parte de sus capas convolucionales al diseñar la red de segmentación [14]. Luego se entrena (o afina) la red completa con el conjunto de datos de segmentación disponible. Esta estrategia permite que el encoder extraiga inicialmente características semánticas útiles y luego el decoder las utilice para producir las máscaras de segmentación. Se ha demostrado que usar pesos preentrenados mejora sustancialmente el desempeño de un U-Net en un conjunto de imágenes aéreas [14].

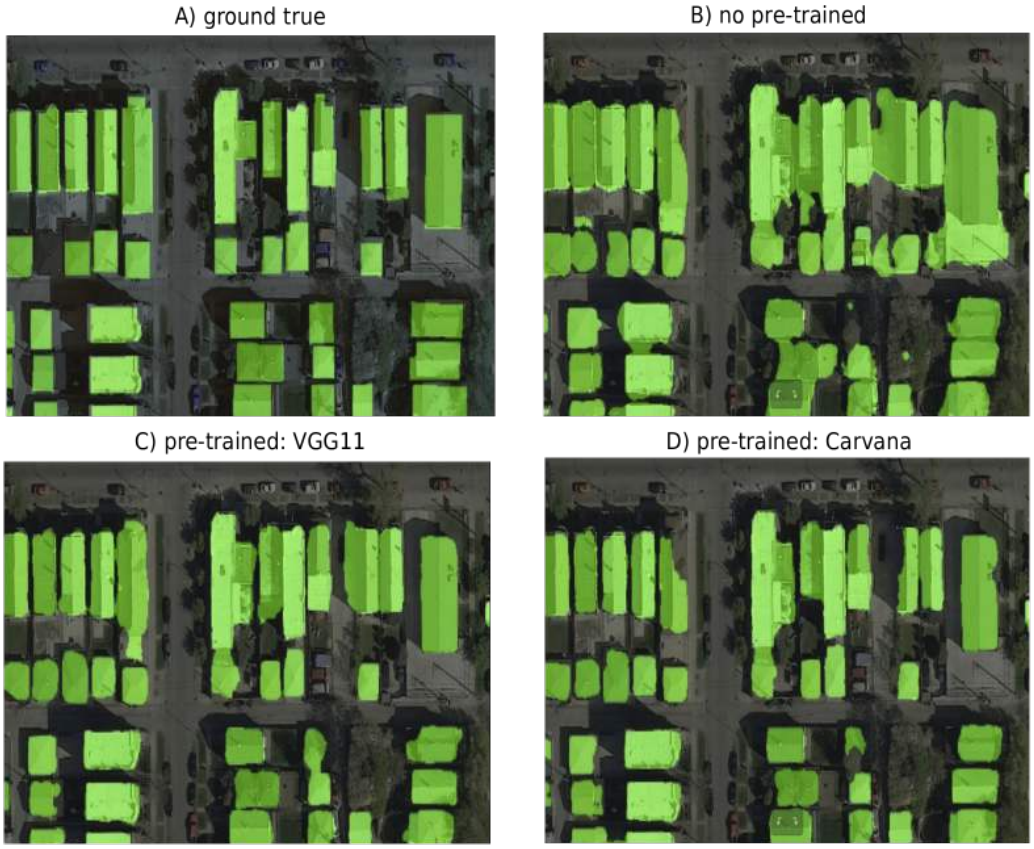


Figura 2.7: Comparación visual del rendimiento en segmentación con diferentes estrategias de inicialización: entrenamiento desde cero (B), encoder preentrenado en ImageNet (C), y red preentrenada en otro dataset (D).

Este enfoque es especialmente común en teledetección: Se presentó un U-Net mejorado cuyo encoder emplea VGG16 preentrenada como extractor de características. Al aplicar esta red a imágenes aéreas de alta resolución, se alcanzaron mejoras notables en la intersección sobre unión (IoU) respecto a un U-Net sin preentrenar [46]. En resumen, el aprendizaje por transferencia permite aprovechar modelos entrenados en grandes bases de datos de clasificación para impulsar la segmentación semántica en dominios con datos limitados, logrando resultados de alta calidad con menos esfuerzo de anotación [14].

2.1.5. Desafíos específicos en segmentación semántica para imágenes aéreas

Desafíos técnicos

La segmentación semántica de imágenes aéreas presenta retos técnicos específicos. Las imágenes capturadas desde drones o satélites suelen tener resoluciones muy altas y gran cobertura espacial, lo que genera volúmenes de datos masivos. Esto obliga a diseñar arquitecturas de redes neuronales y hardware capaz de procesar múltiples gigapíxeles sin sacrificar detalles finos; como se ha señalado, la segmentación de imágenes ultra-altas requiere optimizar eficientemente los recursos de almacenamiento y cómputo para no perder precisión [36]. Reducir la escala de las imágenes puede acelerar el procesamiento, pero a costa de eliminar información crítica y degradar la exactitud de la segmentación

[36]. Otro desafío técnico importante es la computación intensiva: modelos convencionales (como U-Net o DeepLab) pueden agotar rápidamente la memoria de GPU al abordar imágenes enteras de alta resolución. Estudios recientes enfatizan que es necesario equilibrar la complejidad del modelo con la eficiencia del hardware para lograr un rendimiento viable en tiempo real [36]. Además, la obtención de anotaciones pixel-precisas en imágenes aéreas es costosa y laboriosa. A diferencia de conjuntos urbanos a nivel de calle, los datos aéreos carecen de grandes volúmenes de etiquetas de alta calidad; la mayoría de los datasets existentes son limitados en clases y regiones geográficas, lo cual conduce al sobreajuste y limita la generalización de los modelos [37]. Por último, los conjuntos de entrenamiento suelen presentar sesgos (por ubicación geográfica, temporada o tipo de escena) que dificultan la aplicabilidad global. Por ejemplo, en un estudio se observó que los datos etiquetados disponibles provienen de regiones homogéneas con diversidad reducida de geometrías urbanas, condiciones de iluminación y fondos, limitando la validez de los modelos en entornos nuevos [11].

Desafíos visuales

- **Confusión entre clases similares:** En las vistas cenitales, distintos objetos pueden presentar apariencias espectrales o texturales parecidas. Por ejemplo, los tejados de edificaciones y las superficies asfaltadas (calles o estacionamientos) pueden confundirse, especialmente bajo fuertes sombras o con sensores multiespectrales de limitada resolución espectral. Estas similitudes dificultan que la red aprenda a separar contornos con precisión; como se indica en un estudio, la variedad y semántica de los objetos visibles en escenas aéreas complica distinguir entidades con características visuales cercanas [11]. Las sombras pronunciadas y la vegetación (como árboles sobre edificios) pueden además oscurecer bordes y añadir puntos de confusión adicionales.

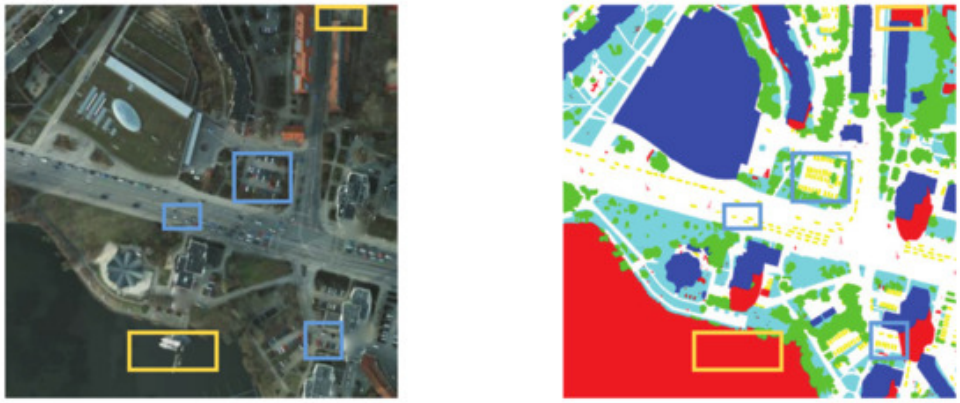


Figura 2.8: Escena del dataset ISPRS Potsdam con tejados y calles de color similar. La imagen izquierda es la vista aérea real y la derecha muestra las etiquetas (con tejados en azul), ilustrando la potencial confusión entre ambos.

- **Oclusiones y obstáculos:** Elementos como nubes, árboles altos o estructuras suplementarias pueden ocultar parcialmente las clases de interés (p.ej., edificios tapados por follaje). Estas oclusiones fragmentan los objetos y rompen la continuidad visual, haciendo difícil la inferencia semántica coherente. Se ha identificado que las oclusiones por contenido superpuesto son un problema destacado, ya que, interrumpen los segmentos y complican la identificación precisa de los límites de los objetos

[11]. Además, la presencia de objetos pequeños o detalles dispersos (vehículos, embarcaciones, antenas) introduce ruido de fondo que exige una segmentación muy fina para no perder información relevante.

Figura 2.9: Dos imágenes con edificios sobre fondos complejos. En la imagen (a) hay calles, árboles, terreno alrededor de estos edificios, y los edificios están situados muy cerca unos de otros. En la imagen (b) los edificios se confunden fácilmente con barcos. Los barcos tienen un tamaño, una textura y una forma similares a los edificios.

- **Variaciones estacionales y angulares:** Las imágenes aéreas pueden adquirirse en diferentes épocas del año o con parámetros de captura distintos. Los cambios de estación alteran drásticamente la apariencia (por ejemplo, vegetación verde vs. otoñal o invernal), mientras que variaciones en el ángulo de vuelo o la inclinación de la cámara modifican la perspectiva geométrica. Tales factores introducen un cambio de dominio entre conjuntos de datos; un modelo entrenado con imágenes invernales puede no funcionar bien en verano, o con sensores distintos. Como se señala en un estudio, las condiciones complejas y cambiantes de adquisición de imágenes (incluyendo iluminación, hora del día y geometría de captura) complican la segmentación, pues el modelo debe ser capaz de generalizar a dominios diversos [37]. En la práctica, esto demanda técnicas de adaptación de dominio o normalización fotométrica para mantener la precisión frente a variaciones contextuales.

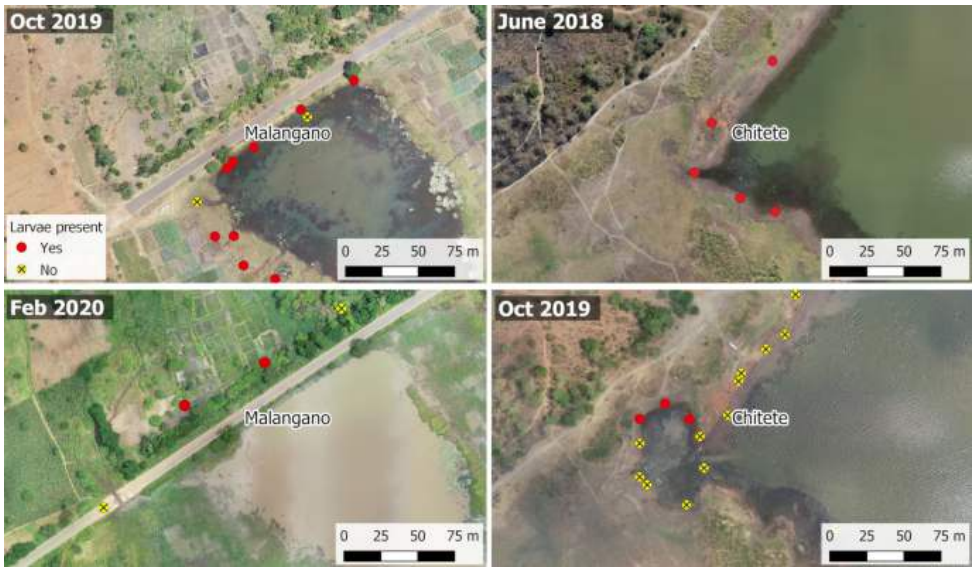


Figura 2.10: Comparación de imágenes aéreas del mismo sitio en distintas estaciones (Izq.: fin de estación seca *vs.* húmeda; Der.: dos épocas secas diferentes). Se aprecian cambios en la vegetación y cuerpos de agua que ilustran la dificultad de la segmentación bajo variaciones estacionales.

Desafíos aplicados

- **Detección de edificios y planificación urbana:** La segmentación semántica es esencial para extraer información espacial detallada sobre construcciones, zonas verdes, vías y otros elementos del paisaje urbano. De esta manera, alimenta sistemas de planificación urbana, catastro digital, gestión del uso del suelo y monitoreo del crecimiento urbano. La precisión requerida es muy alta: por ejemplo, se ha destacado que la extracción exacta de información edificatoria en imágenes de muy alta resolución es un objetivo clave en contextos urbanos [20]. Un error en la delimitación de un edificio puede repercutir en mapas urbanos imprecisos y decisiones erróneas de planificación, por lo que las aplicaciones oficiales exigen métricas de segmentación extremadamente exigentes.

Figura 2.11: Ejemplo de resultados de segmentación semántica sobre vista aérea real: los polígonos coloreados indican edificios detectados. Las delimitaciones son precisas, aptas para planificación urbana.

- **Identificación de construcciones ilegales:** Otro caso de uso crítico es la detec-

ción automática de edificaciones no autorizadas o informales. En estos escenarios, la segmentación semántica debe discriminar sutiles diferencias en áreas densamente edificadas; pequeñas extensiones construidas pueden ser significativas para autoridades de ordenamiento territorial. En consecuencia, es imprescindible minimizar tanto los falsos positivos como los falsos negativos, pues un error puede conllevar sanciones injustas o la falta de intervención oportuna. Este tipo de tarea demanda modelos con sensibilidad y especificidad muy altas, capaces de generalizar en entornos urbanos complejos y cambiantes.

2.1.6. Aplicaciones de la segmentación semántica en imágenes aéreas

La segmentación semántica en imágenes de teledetección asigna una etiqueta de clase a cada píxel (edificio, vía, vegetación, agua, etc.). Esto permite extraer automáticamente elementos urbanos y territoriales de interés. En general, se usa para crear mapas detallados de cobertura del suelo y apoyar la planificación territorial. Por ejemplo, en entornos urbanos es fundamental la detección de edificios: como se indica en un estudio, extraer edificaciones de imágenes de muy alta resolución es "una tarea fundamental" para la construcción y planificación urbanas, así como para la gestión de emergencias [45]. Los autores proponen redes profundas (basadas en U-Net con bloques residuales) que mejoran la precisión al etiquetar píxeles edificados. En la práctica se han usado variantes de U-Net o ResNet para segmentar edificios con gran detalle. Por ejemplo, un trabajo reciente de la Universidad de Alicante entrenó una red ResUNet usando imágenes satelitales (SpaceNet) para extraer máscaras de edificios [40]. Este proyecto demostró aplicaciones concretas: detectar construcciones nuevas a lo largo del tiempo, así como localizar terrenos sin edificaciones (zonas baldías o secas) que podrían destinarse a reforestación o proyectos solares [40]. En resumen, la segmentación semántica permite generar automáticamente mapas de huellas edificadas, útiles tanto para actualizar catastros como para vigilar construcciones ilegales o planificar infraestructuras.

Planificación urbana

En la planificación urbana y el mapeo del espacio, la segmentación semántica automatiza la interpretación de la ciudad. Como apunta un estudio de la PUCP (2018), esta técnica "permite la interpretación automática de los datos" y resulta útil para el mapeo de la cobertura del suelo y la planificación urbana [3]. Con ella es posible extraer no solo edificios, sino también redes viales, parques y otros usos del suelo. Un estudio destaca que la segmentación se ha aplicado extensamente a tareas urbanas: detección de edificios y carreteras, cartografía de uso de suelo y manejo forestal, gracias a la disponibilidad de imágenes VHR [45]. Las clases resultantes (por ejemplo "edificio", "asfalto", "vegetación") facilitan análisis posteriores: identificación de zonas residenciales, delimitación de áreas urbanizadas frente a rurales, diseño de rutas de transporte, etc. Además, iniciativas institucionales españolas ya aplican estas técnicas. Por ejemplo, el proyecto SROADEX (2022) —impulsado por la Infraestructura de Datos Espaciales de España (IDEE) sobre ortofotos PNOA— utiliza IA para segmentar semánticamente las teselas y extraer automáticamente la red viaria nacional [4]. En conjunto, la segmentación semántica se ha convertido en una herramienta clave para generar mapas temáticos urbanos actualizados de forma rápida y precisa.

Monitoreo territorial

Más allá del ámbito urbano, la segmentación semántica se emplea en el monitoreo territorial y ambiental. Por ejemplo, es muy útil en la detección de zonas inundadas a partir de sensores ópticos o drones. Un estudio comparó modelos U-Net, ResNet y DeepLabv3 para segmentar el agua en imágenes aéreas de inundaciones, obteniendo máscaras pixeladas de las zonas anegadas [27]. Este enfoque completamente automatizado aisla las áreas inundadas y genera mapas de inundación con mucha más rapidez que métodos manuales [27], facilitando la respuesta de emergencia y reduciendo pérdidas. De modo similar, la segmentación apoya el monitoreo de recursos naturales: por ejemplo, puede separar categorías de vegetación, bosques y cultivos para el seguimiento de cambios en la cubierta terrestre. Un estudio resaltó que estas técnicas se han usado también en estudios de cobertura de suelo y manejo forestal [45]. En la práctica, redes neuronales han sido entrenadas para clasificar cultivos agrícolas, detectar deforestación o evaluar salud forestal con imágenes satelitales. En resumen, al asignar cada píxel a clases ecológicas (agua, bosque, cultivo, suelo descubierto, etc.), la segmentación semántica facilita un seguimiento detallado del territorio, útil por ejemplo en políticas de conservación, ordenación del agua o agricultura de precisión.

Ejemplos institucionales

Estas aplicaciones no son solo académicas. Instituciones gubernamentales y proyectos institucionales ya emplean segmentación semántica. Además del citado SROADEX para extraer viales en España, en el ámbito internacional empresas tecnológicas usan CNN para generar mapas de edificios (p.ej. retos SpaceNet o DeepGlobe sobre imágenes satelitales). En España, la empresa pública Tracasa Instrumental (Gobierno de Navarra) ha presentado avances en segmentación de edificios usando IA avanzada (difusión generativa) en congresos internacionales (IGARSS 2023). En definitiva, ciudades, administraciones y observatorios territoriales aprovechan la segmentación semántica en imágenes aéreas para detectar construcciones, planificar infraestructuras y monitorear el medio ambiente de forma automatizada [45].

2.2 Arquitecturas de segmentación semántica

En esta sección se presentan las arquitecturas de segmentación semántica utilizadas en este trabajo. Cada una de ellas ha sido seleccionada por su relevancia en el estado del arte, así como por su aplicabilidad al dominio de imágenes aéreas. Se describirá su estructura interna, los componentes clave que definen su funcionamiento, y los mecanismos mediante los cuales logran asignar etiquetas semánticas a nivel de píxel.

Además, se analizarán las ventajas y limitaciones de cada arquitectura, así como las estrategias que emplean para capturar tanto el contexto global de la imagen como los detalles locales necesarios para una segmentación precisa. Esta revisión tiene como objetivo proporcionar una base teórica sólida que justifique la elección de los modelos implementados en el presente proyecto.

2.2.1. DeepLabV3+

DeepLabv3+ es una arquitectura de segmentación semántica de última generación que extiende a DeepLabv3 mediante una estructura codificador-decodificador (*encoder-decoder*) [2]. En esta arquitectura, DeepLabv3 actúa como módulo de codificación de contexto y se añade un módulo decodificador sencillo pero eficaz para refinar los resultados de segmentación, logrando máscaras más definidas, especialmente a lo largo de los bordes de los objetos [2].

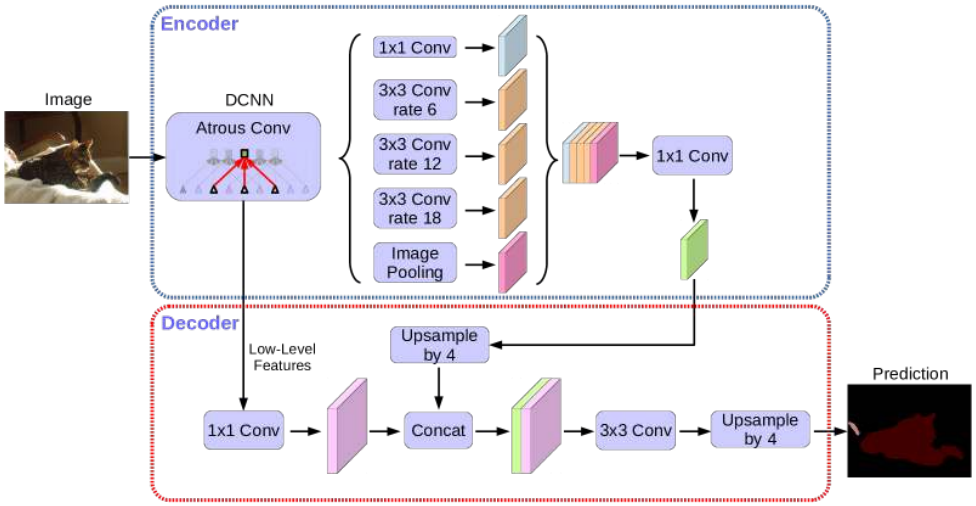
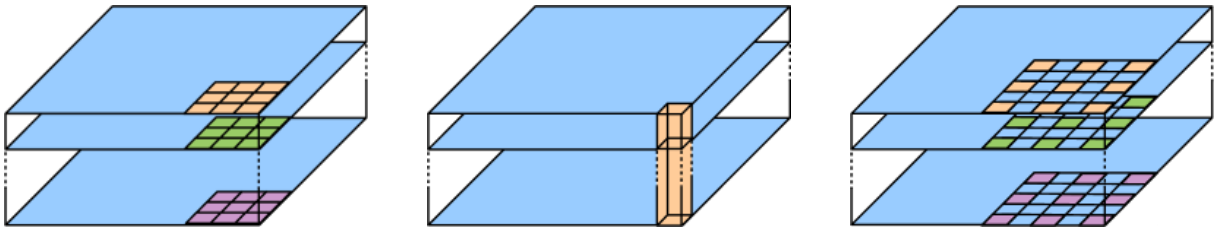


Figura 2.12: Arquitectura de DeepLabV3+. El codificador usa ASPP multiescala y el decodificador refina los bordes.

La clave del encoder de DeepLabv3+ reside en el uso de convoluciones *atrous* (dilatadas) en la red base para preservar una resolución espacial alta en los mapas de características sin incurrir en un coste computacional prohibitivo. En particular, el modelo incorpora un módulo ASPP (*Atrous Spatial Pyramid Pooling*) que aplica convoluciones dilatadas paralelas con distintas *rates* (tasas de dilatación) para capturar información de contexto a múltiples escalas [2]. Además, tanto en el ASPP como en el decodificador se emplean convoluciones separables en profundidad (*depthwise separable convolutions*) en lugar de convoluciones convencionales, reduciendo significativamente el número de parámetros y operaciones manteniendo un alto rendimiento [2]. Este diseño aprovecha los mapas de características multi-escala del encoder y permite que el decoder recupere la información de detalle, obteniendo así una segmentación más precisa en bordes y detalles finos.



(a) Convolución en profundidad

(b) Convolución punto a punto

(c) Convolución dilatada en profundidad (tasa = 2)

Figura 2.13: La convolución separable en profundidad de tamaño 3×3 descompone una convolución estándar en: (a) una convolución en profundidad (aplicando un único filtro por cada canal de entrada), (b) una convolución punto a punto (combinando las salidas anteriores entre canales) y (c) una convolución dilatada en profundidad, en la cual la convolución dilatada se aplica sobre la convolución en profundidad con una tasa de dilatación (*rate*) de 2.

La combinación de amplio contexto multiescala y refinamiento de detalles mediante el decoder hace que DeepLabv3+ sea especialmente adecuado para tareas de segmentación de alta precisión. De hecho, este modelo ha logrado resultados de vanguardia en conjuntos de datos desafiantes, alcanzando por ejemplo un mIOU de 89.0 % en PASCAL VOC 2012 y 82.1 % en Cityscapes *sin* necesidad de posprocesamiento con CRF denso, estableciendo un nuevo estado del arte en su momento [2]. En consecuencia, DeepLabv3+ se considera un referente moderno en segmentación semántica, capaz de delinear con gran fidelidad los contornos de los objetos sin recurrir a técnicas adicionales de refinamiento.

2.2.2. PAN

La *Pyramid Attention Network* (PAN) es una arquitectura de segmentación semántica propuesta en 2018 [19] que combina mecanismos de atención con una pirámide de características para explotar el contexto global de la imagen a la vez que refina los detalles locales. A diferencia de modelos previos como PSPNet (Pyramid Scene Parsing Network) [47], que emplea *pooling* piramidal pero puede perder precisión en la localización de bordes finos debido a dichas operaciones de pooling, o FPN (Feature Pyramid Network) [21], que fusiona características multi-escala en una estructura en U pero sin incorporar explícitamente atención basada en contexto global, PAN busca resolver ambos desafíos de manera conjunta. De hecho, sus módulos especializados —Feature Pyramid Attention (FPA) y Global Attention Upsample (GAU)— están diseñados específicamente para **aumentar el campo de visión efectivo (contexto global) y recuperar detalles de localización espacial** en las predicciones [19], afrontando así el problema de segmentar objetos de diversos tamaños sin sacrificar la precisión en contornos y regiones pequeñas.

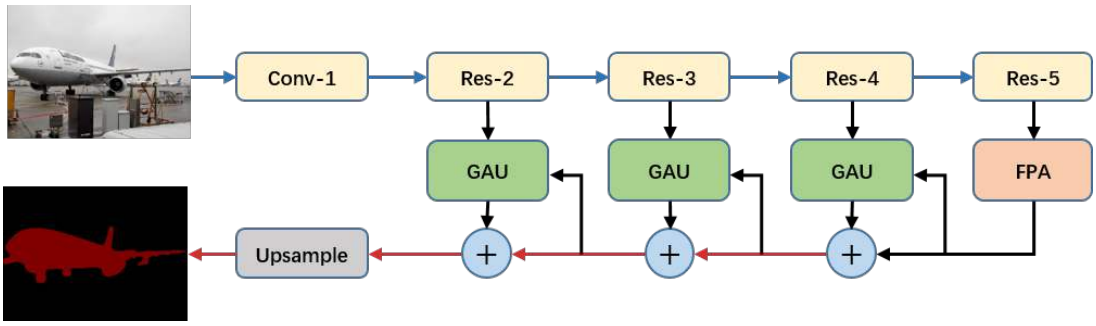
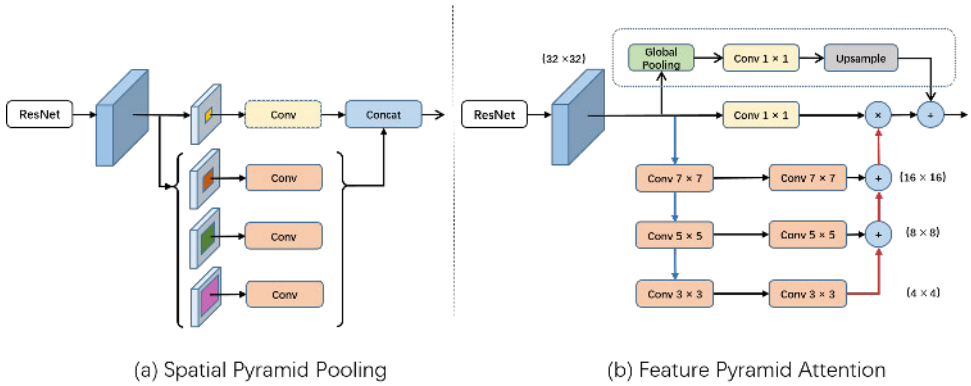


Figura 2.14: Arquitectura completa de Pyramid Attention Network (PAN). Se muestra la integración del módulo Feature Pyramid Attention (FPA) entre el encoder y el módulo Global Attention Upsample (GAU) en el decoder.

El **módulo Feature Pyramid Attention (FPA)** actúa como bloque de cuello de botella al final del *encoder* de PAN. Este módulo realiza una atención espacial piramidal sobre la característica de alto nivel extraída por la red base (p. ej., ResNet-101 dilatada) [19]. Consta de una serie de sub-bloques con diferentes escalas de filtrado (típicamente convoluciones de 7×7, 5×5 y 3×3) que se organizan en una estructura en forma de U, inspirada en la FPN, para fusionar jerárquicamente la información de contexto a tres escalas. Debido a que la resolución de las características de alto nivel es reducida (por ejemplo 1/16 de la imagen), el uso de kernels grandes (como 7×7) no supone un costo prohibitivo. La integración piramidal se realiza paso a paso, combinando primero las características de la escala más pequeña con la siguiente, y así sucesivamente, lo que permite incorporar eficientemente el contexto de vecinos de escala de manera precisa. Finalmente, el resultado de esta pirámide de atención se proyecta mediante una convolución 1×1 y se utiliza como un mapa de atención espacial que modula (mediante multiplicación elemento a elemento) la característica original de alto nivel del *encoder*. Además, FPA incluye una rama de *pooling* global que extrae un vector de contexto global (mediante *average pooling* de toda la característica) el cual se suma a las características salientes del módulo. Este componente global refuerza la representación con información de toda la escena, mejorando la capacidad de la FPA para destacar las regiones de interés relevantes en el mapa de características. En conjunto, la FPA **fusiona información de múltiples escalas con contexto global para producir una atención pixel-wise más precisa** sobre las características de alto nivel, lo que ayuda a identificar correctamente incluso objetos pequeños o difíciles mediante un mayor campo receptivo efectivo.



(a) Spatial Pyramid Pooling (b) Feature Pyramid Attention

Figura 2.15: Estructura del módulo Feature Pyramid Attention (FPA). La imagen muestra las ramas de convolución piramidal y la integración del pooling global para generar atención espacial multi-escala.

Por su parte, el **módulo Global Attention Upsample (GAU)** se encarga de la fase de *decoder*, guiando la fusión de las características de bajo nivel (más resoluciones altas) con la información semántica global proveniente de niveles superiores. A diferencia de decodificadores tradicionales que simplemente concatenan o suman mapas de distintas resoluciones (como en una FPN clásica) o usan upsampling ingenuo, GAU introduce un mecanismo de atención global en cada paso de upsampling. En concreto, para una determinada etapa del decoder, GAU toma la característica de nivel alto (por ejemplo, la salida procesada por FPA u otra capa más arriba en la pirámide) y calcula a partir de ella un vector de contexto global mediante *global average pooling*. Este vector pasa por una pequeña capa densa (convolución 1×1 con normalización por batch y activación ReLU) para producir un conjunto de pesos de atención. Mientras tanto, la característica de nivel inferior (p. ej. una feature map de una etapa anterior del encoder, con mayor resolución) es sometida a una convolución (usualmente 3×3) para reducir su dimensionalidad. El vector de contexto global obtenido de la característica de alto nivel se utiliza entonces para modular la característica de nivel inferior: se realiza una multiplicación canal a canal entre ambos, actuando el vector global como un filtro de atención que realza o atenua las respuestas locales en función del contexto de la escena. Tras esta modulación, la característica de alto nivel (semánticamente rica pero más cruda en detalle) se **suma** a la característica de bajo nivel ya ponderada, combinando así la información global con los detalles locales. El resultado de esa suma es entonces **upsampleado** (por ejemplo, duplicando la resolución) para servir de entrada a la siguiente etapa del decoder. Este procedimiento se repite en cascada a través de las distintas escalas (capas del decoder), propagando progresivamente la guía del contexto global hacia las resoluciones más finas. Gracias a su diseño sencillo pero eficaz, la GAU logra aprovechar los *feature maps* de distintos niveles de forma más efectiva, usando las características de alto nivel para **guiar la selección de detalles locales** relevantes de una manera computacionalmente ligera. En suma, GAU actúa como un mecanismo de atención global en el proceso de reconstrucción, afinando bordes y detalles al mismo tiempo que mantiene la coherencia contextual de las etiquetas a nivel de píxel.

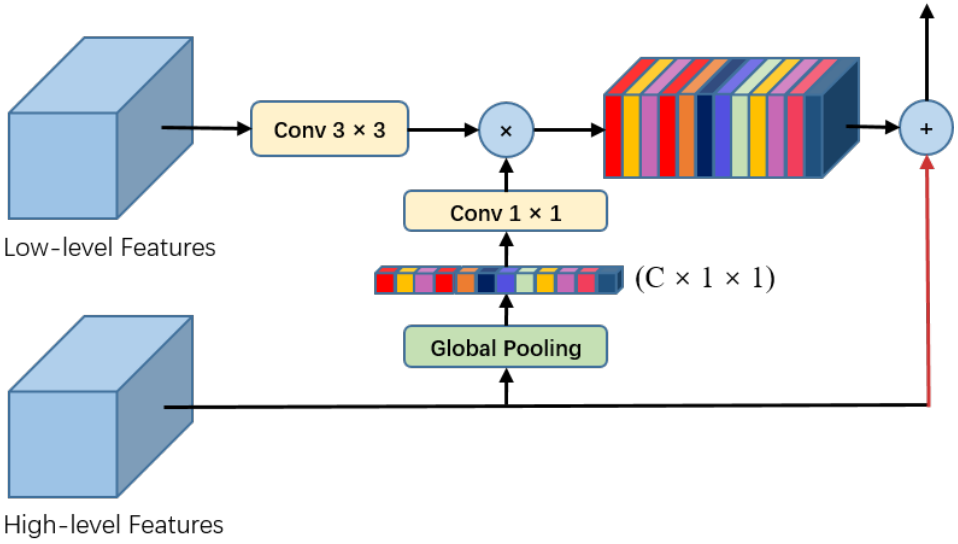


Figura 2.16: Estructura del módulo Global Attention Upsample (GAU). El módulo genera atención global desde características profundas para guiar el refinamiento de detalles espaciales en la segmentación.

La arquitectura completa de PAN integra la FPA como bloque central entre el encoder y el decoder, y aplica módulos GAU en las sucesivas fases de upsampling. Este enfoque le permite superar las limitaciones de PSPNet y otras arquitecturas previas: por un lado, **aprovecha explícitamente el contexto global** de la imagen (a través del pooling global y la atención piramidal) mejorando la clasificación de píxeles en regiones ambiguas o en objetos pequeños; y por otro, **refina la información espacial** mediante la retroalimentación de detalles locales guiada por dicha información global, evitando la pérdida de resolución típica de arquitecturas puramente piramidales sin decoder dedicado. A diferencia de PSPNet o DeepLab, que realizan la segmentación final con mapas de características de baja resolución simplemente re-escalados y pueden omitir bordes finos, PAN recupera las estructuras detalladas al combinar las capas bajas con las altas de forma informada. Asimismo, a diferencia de decodificadores complejos propuestos en otros trabajos (p. ej. múltiples capas de convolución transpuesta), GAU proporciona una solución más simple y eficiente para la fusión multinivel, ya que introduce mínimas operaciones adicionales al tiempo que mejora la precisión de los contornos. Gracias a estas innovaciones, PAN ha demostrado un rendimiento sobresaliente: los autores reportan resultados *state-of-the-art* en conjuntos de datos desafiantes como PASCAL VOC 2012 y Cityscapes, alcanzando por ejemplo un **mIoU de 84.0 %** en PASCAL VOC 2012 sin utilizar datos de pre-entrenamiento adicionales de COCO. Esto confirma la eficacia de combinar un **contexto global robusto con la refinación de detalles locales** en una arquitectura de segmentación semántica de última generación.

2.2.3. UPerNet

La arquitectura *UPerNet* fue propuesta para abordar la tarea de *Unified Perceptual Parsing*, es decir, la interpretación unificada de la escena a múltiples niveles semánticos [41]. Esta tarea busca que un único modelo reconozca simultáneamente tantos conceptos visuales como sea posible en una imagen, abarcando la clasificación de la escena completa, la segmentación de los objetos que contiene, la identificación de las partes de esos

objetos, así como la segmentación de materiales y la clasificación de texturas [41]. Para lograr integrar todas estas tareas de distinta granularidad espacial y semántica, UPerNet adopta un diseño de red multiescala y jerárquico [41], inspirado en arquitecturas previas de segmentación semántica como *Feature Pyramid Network* (FPN) [21] y *Pyramid Pooling Module* (PPM) de PSPNet [47].

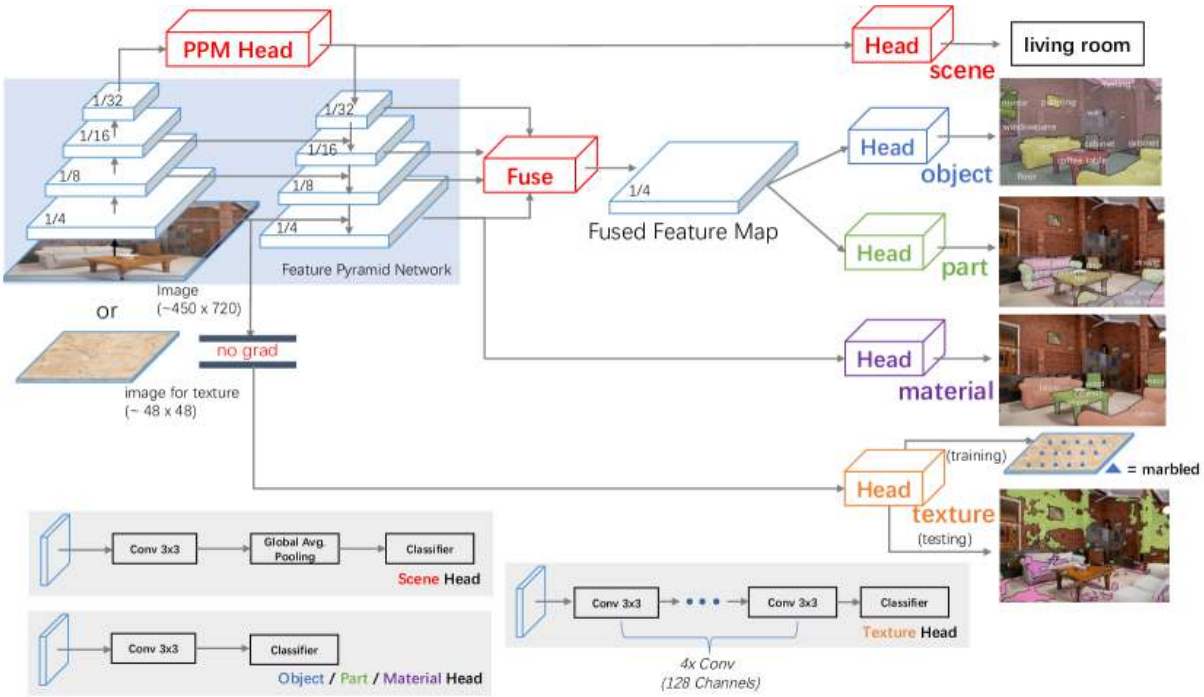


Figura 2.17: Arquitectura de UPerNet, combinando un FPN con un módulo de Pyramid Pooling (PPM).

En esencia, UPerNet extiende la idea de FPN incorporando un módulo de pooling piramidal (PPM) para capturar información de contexto global [41]. FPN provee un *pipeline* de arriba-abajo (*top-down*) con conexiones laterales que combina las representaciones de múltiples niveles de una red profunda, produciendo mapas de características a diversas escalas (p.ej. P2, P3, P4, P5) [21]. De este modo, las características de nivel alto (con semántica global) se fusionan con las de nivel bajo (de mayor resolución espacial), enriqueciendo los detalles finos con el contexto de alto nivel [21, 41]. UPerNet primero aplica el PPM sobre la característica de mayor profundidad (salida final del extractor de características) antes de iniciar la fase top-down de FPN [41]. El PPM realiza pooling a múltiples escalas sobre dicha característica, obteniendo una representación global de la escena que posteriormente se combina con las características piramidales [47, 41]. Esto permite incorporar *a priori* globales efectivos que mitigan la limitada cobertura del campo receptivo efectivo de una CNN profunda estándar [41]. La salida del PPM se utiliza como mapa de características inicial (nivel P5) en la pirámide FPN, a partir del cual se generan las características de niveles más finos (P4, P3, P2) mediante la ruta descendente con fusiones laterales [41, 21]. Los autores reportan que la inclusión del PPM es altamente compatible con la estructura FPN y produce mejoras significativas en el rendimiento de segmentación al aportar contexto global [41].

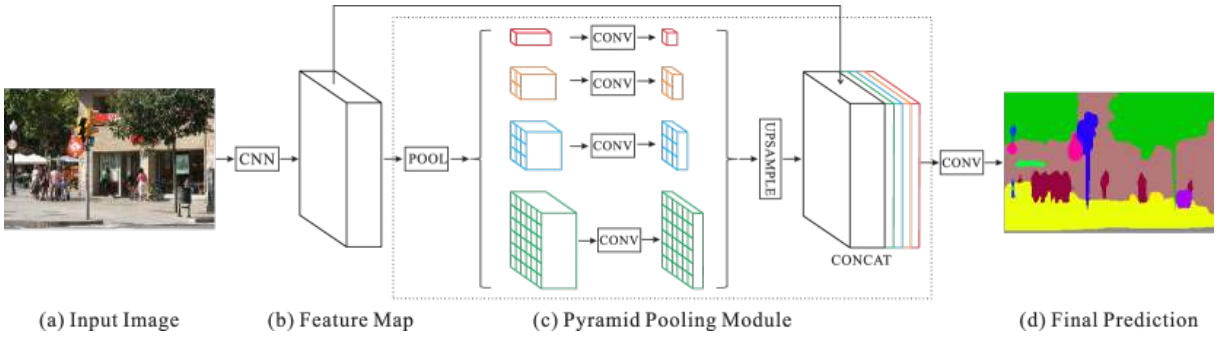


Figura 2.18: Esquema de PSPNet con su **Pyramid Pooling Module**. Este módulo extrae información de contexto global en múltiples escalas, que luego se fusiona con el mapa de características original.

Una vez obtenidas las características multiresolución, UPerNet añade *cabezas* de predicción especializadas en distintos niveles de la pirámide, alineando cada sub-tarea con el nivel semántico apropiado [41]. En particular, una cabeza de clasificación de escena se conecta al mapa de características de resolución más baja pero semántica más alta (P5, justo después del PPM), aprovechando la información de toda la imagen para predecir la categoría de la escena [41]. Para la segmentación semántica de objetos (y la segmentación de partes de objetos), UPerNet fusiona las características de *todos* los niveles de la pirámide (P2, P3, P4, P5) y emplea esta característica combinada como entrada de las cabezas de objeto y de partes [41]. Esta fusión multiescala provee simultáneamente contexto global y detalles locales, resultando adecuada para distinguir objetos de distintas escalas y desambiguar las partes dentro de cada objeto [41]. De hecho, se observó que utilizar la combinación de todos los mapas de FPN supera a usar solo el mapa de mayor resolución, mejorando la precisión en ambas tareas [41]. Por otro lado, la segmentación de materiales (superficies) se realiza a partir del mapa de características de más alta resolución en FPN (P2), dado que la identificación de materiales depende de texturas y detalles locales finos que se preservan mejor en esa escala [41]. Finalmente, la predicción de texturas (un atributo de muy bajo nivel) se desacopla de la pirámide principal y se lleva a cabo usando las características de las primeras capas convolutivas de la red (antes de FPN), ya que las texturas se reconocen mediante patrones locales de baja complejidad [41]. Esta rama de texturas se entrena por separado al final, para no interferir con las representaciones internas necesarias para las otras tareas [41].

En resumen, UPerNet logra unificar en una sola red el *parsing* semántico completo de la escena a múltiples escalas. Gracias a su enfoque jerárquico multiescala, la arquitectura puede aprovechar información compartida entre tareas y proporcionar contexto de alto nivel durante las predicciones de bajo nivel [41]. Esto le permite abordar eficazmente tareas complejas de interpretación de escenas (*scene parsing* completo, englobando escena + objetos + partes) dentro de un marco unificado, imitando en cierto modo la forma en que los humanos entienden simultáneamente el contexto global de una escena y sus detalles locales [41].

2.2.4. U-Net

La arquitectura U-Net es una red neuronal convolucional de segmentación caracterizada por su estructura en forma de "U" simétrica, compuesta por un módulo codificador

(etapa de contracción) y un módulo decodificador (etapa de expansión) conectados mediante *skip connections* (conexiones de salto) [34]. El brazo codificador reduce progresivamente la resolución espacial de la imagen mediante convoluciones seguidas de operaciones de *pooling*, extrayendo características de alto nivel y capturando el contexto global de la escena [34]. Por su parte, el brazo decodificador invierte este proceso: realiza *upsampling* (p. ej., mediante convoluciones transpuestas) y fusiona las características detalladas provenientes del codificador a través de las conexiones de salto, restaurando la información fina que se había perdido durante la contracción [34]. De esta manera, la U-Net logra combinar información de contexto con detalles locales: la ruta de contracción provee contexto para entender "qué" hay en la imagen, mientras que la ruta expansiva junto con las *skip connections* asegura "dónde" se encuentra cada objeto, permitiendo una localización precisa de estructuras y bordes en la segmentación resultante [34].

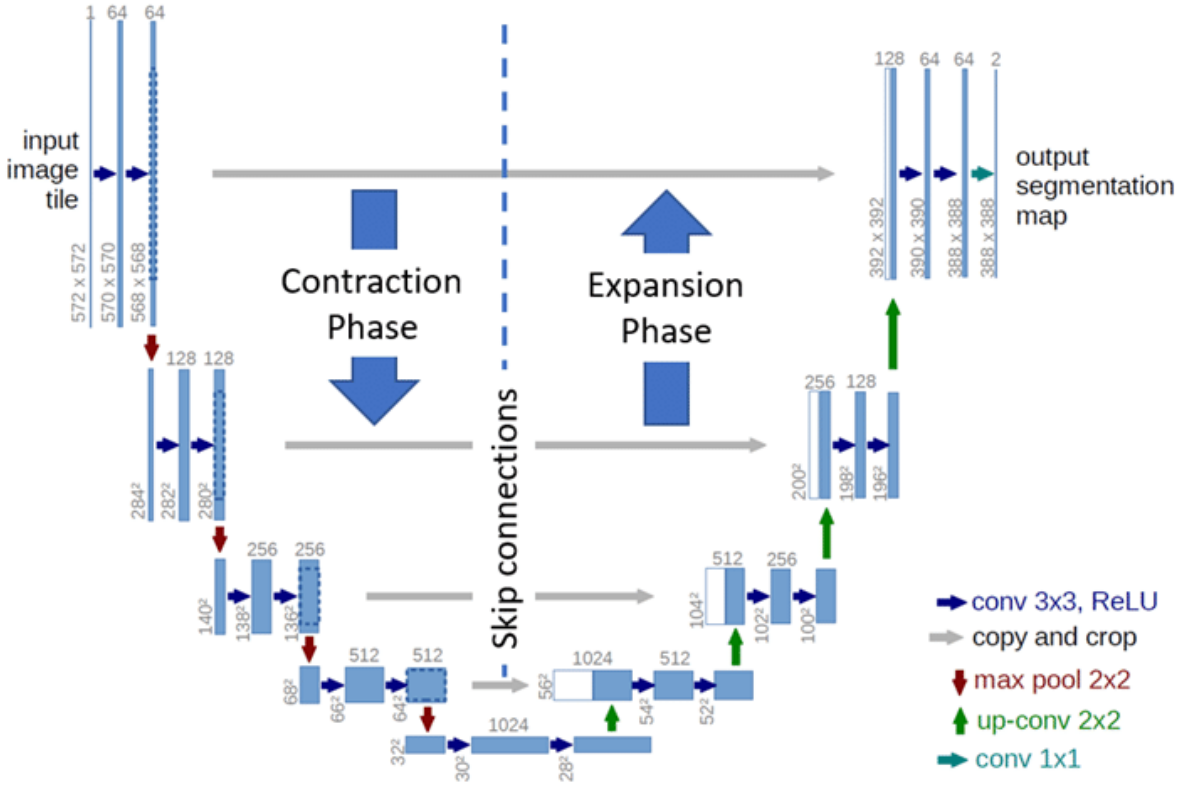


Figura 2.19: Arquitectura básica de U-Net (contracción-expansión con *skip connections*).

Originalmente, U-Net fue propuesta para la segmentación de imágenes biomédicas [34]. Sin embargo, gracias a su eficacia y diseño generalizable, se ha convertido en una de las arquitecturas más utilizadas para la segmentación semántica en múltiples dominios [16]. De hecho, U-Net es ampliamente empleada en ámbitos como la teledetección y las imágenes aéreas para tareas como la clasificación de cobertura del suelo, detección de carreteras o extracción de edificios, demostrando su aplicabilidad más allá del sector médico [32]. Gracias a su equilibrio entre precisión y simplicidad, U-Net ha tenido un gran impacto en visión por computador. Ha inspirado numerosas variantes y sigue siendo un modelo de referencia habitual en aplicaciones prácticas y competencias de segmentación semántica.

2.3 Backbones

Los *backbones* constituyen los extractores de características visuales en las arquitecturas de segmentación semántica, actuando como la base del encoder. En este trabajo se utilizan diversos modelos preentrenados, todos ellos entrenados inicialmente en el conjunto de datos **ImageNet**.

El preentrenamiento en ImageNet permite que los backbones aprendan patrones visuales generales —como bordes, texturas y formas— que luego pueden transferirse a tareas específicas, como la segmentación semántica aérea, mediante técnicas de aprendizaje por transferencia (fine-tuning). Este enfoque mejora la convergencia del entrenamiento y la capacidad de generalización en datasets especializados y de tamaño medio.

A continuación se describen los modelos seleccionados, destacando su estructura, principios de diseño y aportaciones al rendimiento del sistema global.

Todos estos backbones han sido preentrenados en ImageNet y optimizados para extraer representaciones visuales profundas de forma eficiente, equilibrando precisión y costes computacionales, lo que los hace particularmente adecuados para segmentación semántica de imágenes aéreas.

2.3.1. ResNeXt101_32x8d

ResNeXt101_32x8d es una arquitectura de red neuronal convolucional residual de 101 capas que introduce el concepto de cardinalidad en sus bloques residuales agregados [42]. Esta arquitectura, sigue un diseño altamente modular y de múltiples ramas (multi-branch): consiste en la repetición de un bloque básico con varias ramas paralelas a lo largo de la red, con solo unos pocos hiperparámetros a ajustar. La filosofía de ResNeXt enfatiza aumentar el número de caminos independientes (cardinalidad) dentro de cada bloque residual en lugar de simplemente hacer la red más profunda o más ancha. De hecho, manteniendo constante la complejidad (parámetros y FLOPs), incrementar la cardinalidad demostró mejorar la precisión de clasificación de forma más efectiva que incrementar la profundidad o la amplitud de la red [42]. En otras palabras, la cardinalidad se propone como un factor fundamental (junto con la profundidad y la anchura) para escalar la capacidad del modelo.

Cada bloque residual en ResNeXt101_32x8d adopta la estructura tipo bottleneck de tres capas (conv 1×1 → conv 3×3 → conv 1×1), pero con una modificación clave: la convolución central de 3×3 se realiza mediante una convolución agrupada en 32 grupos. Es decir, en vez de una sola convolución monolítica, el bloque divide la operación en 32 subconjuntos de filtros que actúan en paralelo sobre diferentes porciones de los mapas de características [42]. Cada uno de estos 32 caminos independientes procesa una representación intermedia de baja dimensionalidad (obtenida gracias a la reducción de canales por la conv 1×1 inicial) y aplica una transformación convolucional 3×3 sobre ese sub-espacio. El resultado de estas 32 transformaciones se agrega por suma al final del bloque (en lugar de concatenarse), combinándose para formar la salida del bloque residual antes de la conexión de atajo (skip connection). Esta estrategia de dividir-transformar-combinar mediante suma asegura que la salida del bloque tenga la misma dimensionalidad que cada

rama individual, evitando un aumento de dimensionalidad al fusionar las ramas.

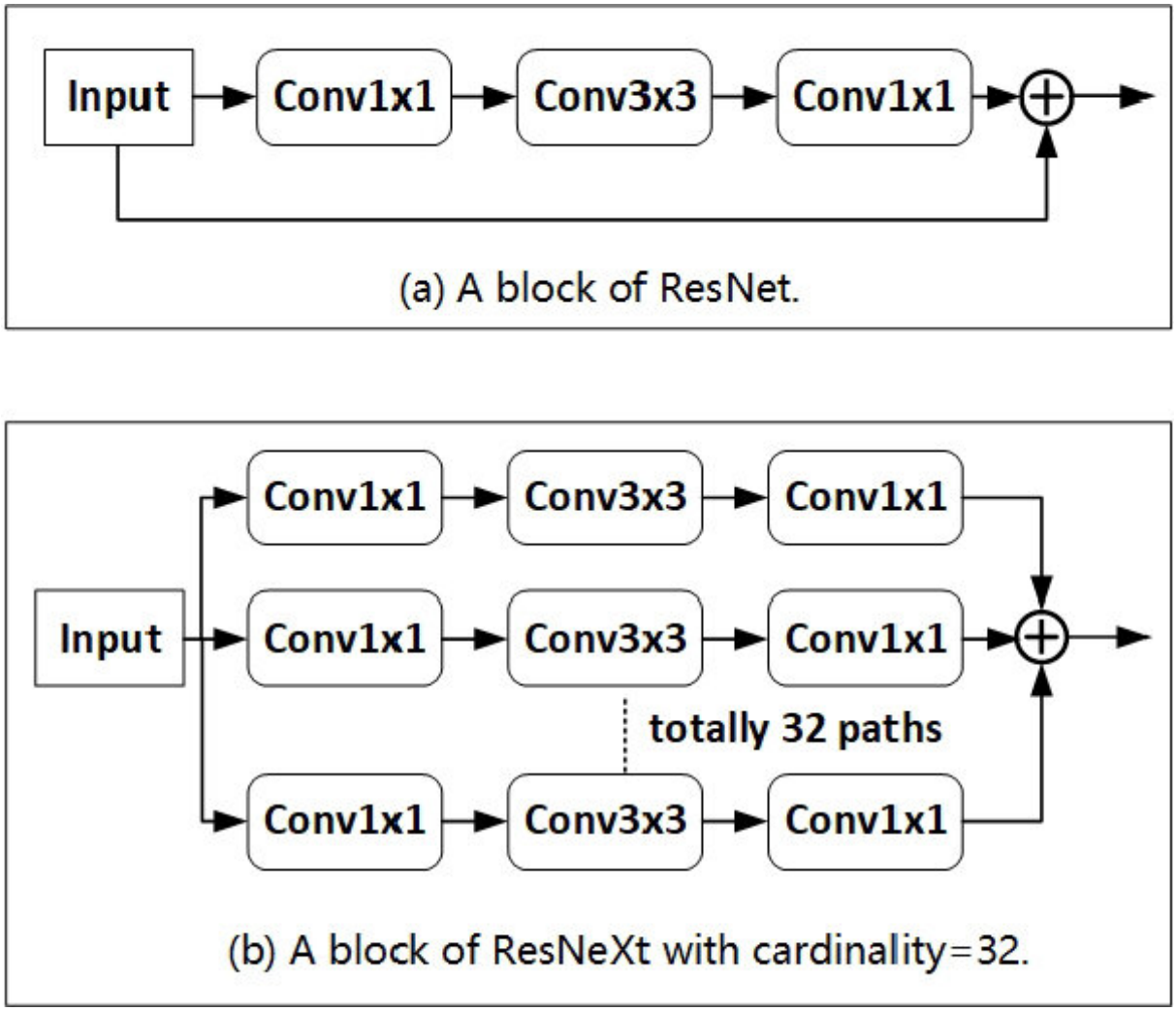


Figura 2.20: Bloque residual: comparación entre ResNet (arriba) y ResNeXt con convoluciones agrupadas (abajo).

La notación ResNeXt101_32x8d resume los hiperparámetros principales de este modelo: 101 indica la profundidad de la red (101 capas entrenables en total), mientras que 32x8d refiere a la cardinalidad = 32 junto con un ancho de bloque de 8 filtros (por cada rama en la convolución agrupada) [9]. En otras palabras, cada bloque residual agrupa sus convoluciones en 32 caminos paralelos (cardinalidad 32) y cada uno de esos caminos opera con un subtensor de 8 filtros (de ahí "8d"). Esto da como resultado que la convolución agrupada 3×3 de cada bloque maneja efectivamente $32\times8 = 256$ filtros distribuidos en 32 grupos, en lugar de 256 filtros densamente conectados; gracias a la estructura bottleneck, primero se reducen los canales de entrada (con la conv 1×1 inicial) para que cada grupo opere sobre menos canales, y al final del bloque otra conv 1×1 restaura la dimensionalidad original (expandiendo los canales nuevamente).

En cuanto a la organización de las capas, la red se divide típicamente en cuatro etapas secuenciales (después de una capa inicial de conv 7×7 + *pooling*): estas etapas contienen 3, 4, 23 y 3 bloques residuales bottleneck, respectivamente, sumando un total de 33 bloques a lo largo de la arquitectura. Cada bloque aporta 3 capas convolucionales, y junto con

la capa inicial suman aproximadamente las 101 capas (contando todas las convoluciones, 101 es el número de capas ponderadas) del modelo. Las transiciones entre etapas se realizan aumentando el *stride* en la primera conv 1×1 de ciertos bloques para reducir la resolución espacial (similar al esquema de ResNet), manteniendo un diseño jerárquico donde la profundidad aumenta en cada etapa.

Un aspecto destacado de ResNeXt101_32x8d es su eficiencia computacional dada la combinación de diseño modular y convoluciones agrupadas. Al repartir una misma cantidad de filtros en múltiples grupos paralelos, cada grupo aprende una transformación más pequeña con menos parámetros, manteniendo aproximadamente constante el número total de parámetros y operaciones del bloque residual frente a un bloque convencional equivalente [42]. Esto permite que ResNeXt aumente la capacidad del modelo (mediante más ramas paralelas) sin un costo computacional prohibitivo. Además, como todas las ramas dentro de un bloque tienen la misma topología, la implementación mediante convolución agrupada aprovecha la paralelización: las 32 convoluciones en grupo pueden ejecutarse simultáneamente de forma eficiente en una única operación optimizada. En resumen, el backbone ResNeXt101_32x8d ofrece un diseño residual de múltiples ramas escalable, donde la cardinalidad actúa como factor clave para lograr mayor poder de representación de forma eficaz y con un costo computacional controlado [42]

2.3.2. ConvNeXt_Base

ConvNeXt_Base es un modelo perteneciente a la familia **ConvNeXt**, diseñada en 2022 como una reimaginación moderna de las CNN tradicionales [22]. La motivación del trabajo, titulado *"A ConvNet for the 2020s"*, fue demostrar que una arquitectura puramente convolucional, debidamente refinada, puede rivalizar o superar a los Vision Transformers en tareas de reconocimiento visual, incluidas las de detección y segmentación semántica, conservando la simplicidad y eficiencia de las CNN clásicas.

ConvNeXt_Base hereda el marco modular de ResNet, pero incorpora múltiples mejoras inspiradas en los Transformers. Entre estas se incluyen:

- **Stem simplificado**: reemplazo de la clásica conv 7×7 + pooling por una única conv 4×4 con stride 4, reduciendo abruptamente la dimensionalidad inicial.
- **Bloques regulados por Layer Normalization**: se sustituyó el BatchNorm por LayerNorm ubicado después de la convolución depthwise, seguido de una conv 1×1, activación GELU, y otra conv 1×1, replicando la estructura de Transformer pero manteniendo elementos convolucionales.
- **Aumento del tamaño del kernel y strides ajustados**: en los bloques deeper se emplean strides mayores y kernels ampliados, lo que permite una mayor capacidad de modelado espacial con un coste computacional moderado.

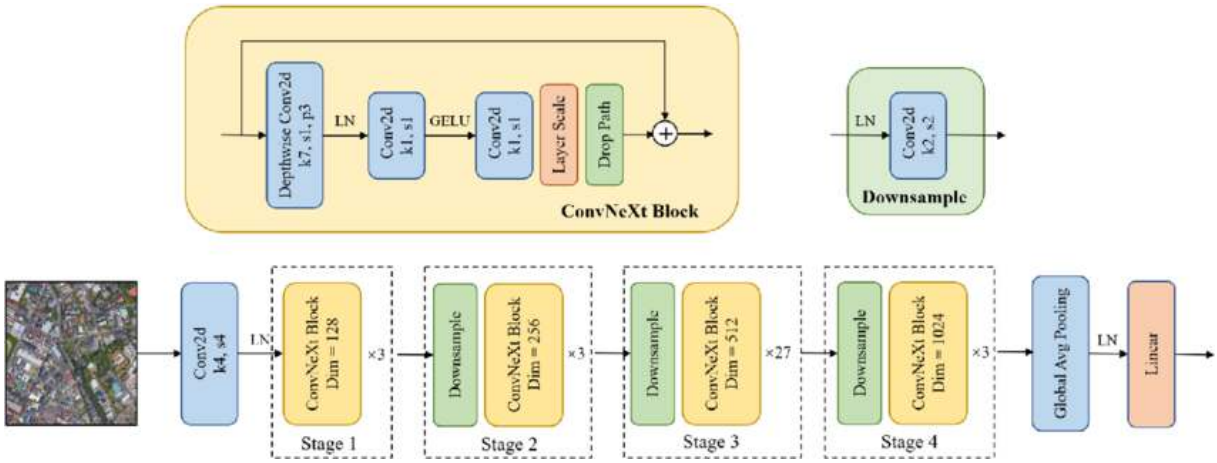


Figura 2.21: Arquitectura de ConvNeXt, una jerarquía de características de cuatro etapas, que se construyó para extraer características de diferentes escalas. La capa downsample y el bloque ConvNeXt se apilan en cada etapa en la proporción 3:3:27:3. LN y GELU representan el Layer Normalization y la activación GELU.

El modelo Base dispone de unos 88M de parámetros y 15.4GFLOPs, lo que resulta en una precisión **Top-1 del 84.06 % en ImageNet-1K** (sin técnicas de poda). Además, ConvNeXt_Base ha demostrado superar a arquitecturas como Swin Transformer en benchmarks de detección (COCO) y segmentación semántica (ADE20K), destacando su capacidad como backbone moderno y eficiente.

Por su arquitectura optimizada y rendimiento sobresaliente, ConvNeXt_Base se presenta como un candidato ideal para tareas de segmentación semántica en imágenes aéreas, ofreciendo alto poder representacional sin verse penalizado por sobrecostes computacionales.

2.3.3. ResNet34

ResNet34 forma parte de la familia de redes residuales propuestas en 2016 bajo el título *"Deep Residual Learning for Image Recognition"* [12]. Su aporte central es la introducción de **bloques residuales**, donde las capas aprenden funciones residuales ($F(x) = H(x) - x$) en lugar de aprender directamente el mapeo $H(x)$. Las conexiones de identidad (skip-connections) permiten que los gradientes fluyan más fácilmente a través de redes profundas, facilitando el entrenamiento y mitigando el problema de la degradación en redes muy profundas.

Las principales características de ResNet34 incluyen:

- **Bloque residual básico**: cada bloque consta de dos capas convolucionales de 3×3 con BatchNorm y activación ReLU, seguido por una conexión de identidad que suma la entrada original a la salida de las convoluciones.
- **Profundidad de 34 capas**: la red se organiza en cuatro etapas, con 3, 4, 6 y 3 bloques residuales respectivamente, lo que junto a la capa inicial de conv 7×7 + pooling y la capa final de clasificación da el total de 34 capas ponderadas.

- **Eficiencia computacional**: con aproximadamente 21.8 millones de parámetros y 3.6 GFLOPs por imagen 224×224, ResNet34 ofrece un equilibrio notable entre precisión y eficiencia.
- **Rendimiento en ImageNet**: alcanza cerca del 73.3 % de precisión Top-1 y 91.4 % Top-5 en ImageNet-1K usando un pipeline estándar de entrenamiento.

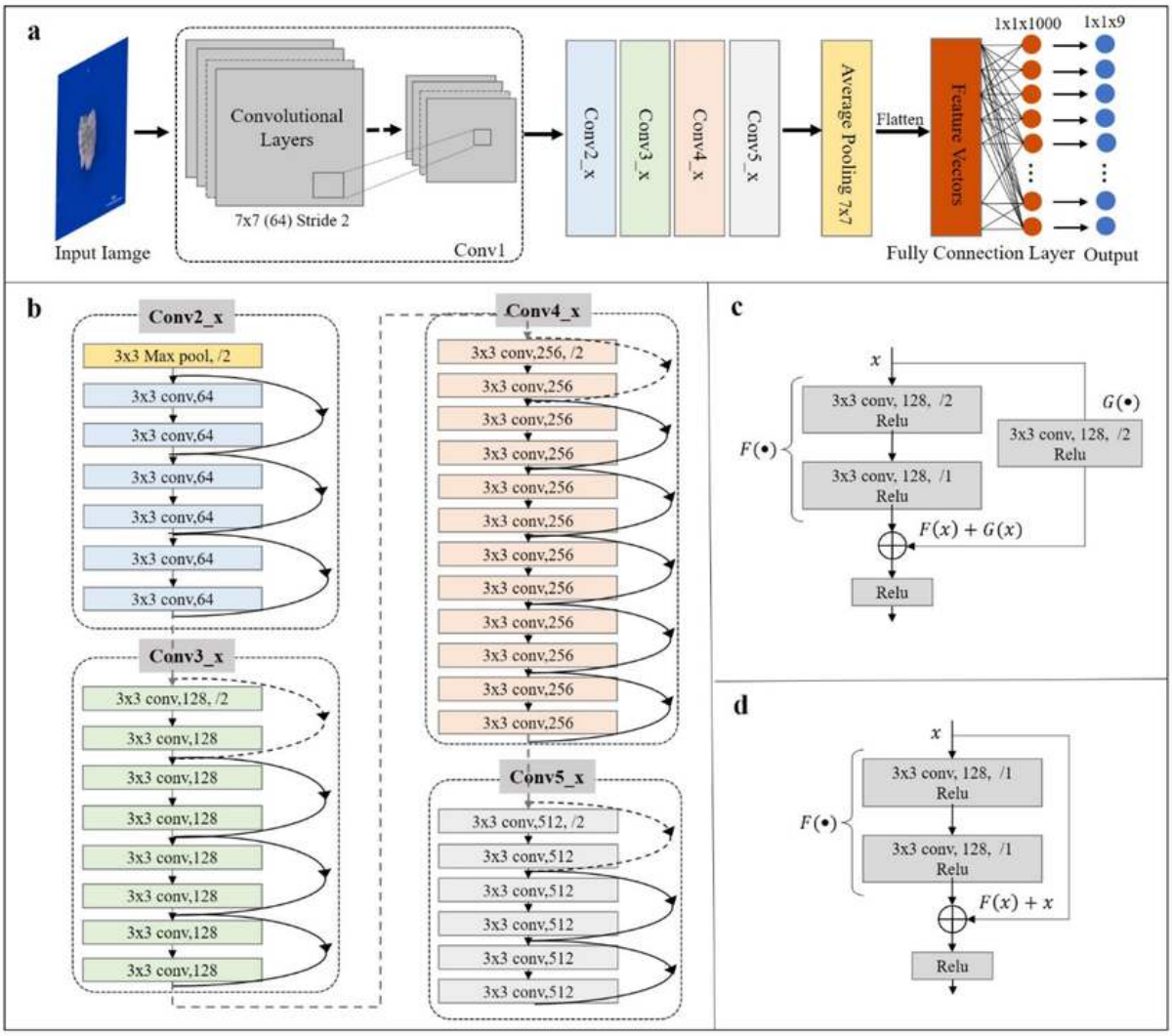


Figura 2.22: Arquitectura general de ResNet-34: etapas Conv2_x a Conv5_x con bloques residuales.

Gracias a su arquitectura modular y eficiente, ResNet34 es una opción frecuente como **backbone** en tareas de visión por computador, incluidas la segmentación semántica y la detección, especialmente cuando se busca un buen balance entre rendimiento y uso de recursos.

ResNet34 se emplea habitualmente como extractor de características en arquitecturas encoder-decoder y se pre-entrena sobre ImageNet para posteriormente afinarse (fine-tuning) en datasets específicos, lo que le permite generalizar bien en diversas tareas con cantidades moderadas de datos anotados.

2.3.4. MobileNetV2

MobileNetV2, presentado en 2018 en *"Inverted Residuals and Linear Bottlenecks"*, surge como una arquitectura eficiente destinada a dispositivos móviles, optimizada para tareas como clasificación de imágenes, detección y segmentación semántica [35]. Su principal innovación es el bloque residual invertido con *linear bottleneck*, donde los atajos (skip connections) se aplican entre capas comprimidas, en lugar de entre capas expandidas como en ResNet, lo que reduce el uso de memoria y mejora el flujo de gradientes.

Además, emplea convoluciones separables en profundidad (*depthwise separable convolutions*), que descomponen una convolución estándar en una convolución *depthwise* más una *pointwise* (1×1), reduciendo entre 8 y 9 veces el coste computacional manteniendo un rendimiento comparable.

Los bloques invertidos siguen la forma: expansión (1×1 Conv + ReLU6), convolución depthwise 3×3 (filtrado espacial) y proyección lineal (1×1 Conv sin activación), garantizando la preservación de información en espacios reducidos.

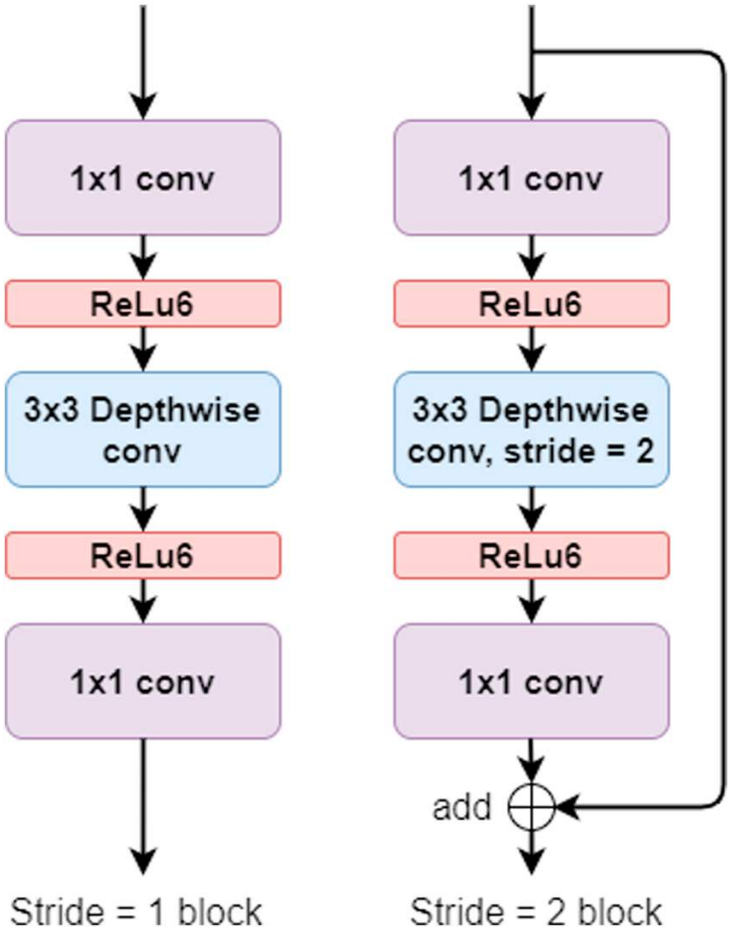


Figura 2.23: Bloque invertido con *linear bottleneck* en MobileNetV2: expansión (1×1), convolución depthwise 3×3, proyección lineal (1×1) y conexión residual entre las capas comprimidas.

MobileNetV2 con factor de anchura 1.0 cuenta con aproximadamente 3.5 millones de parámetros y 300 MFLOPs, alcanzando una precisión Top-1 del 72.2 % y Top-5 del 90.3 %

en ImageNet-1K.

Gracias a su diseño compacto y eficiencia, MobileNetV2 se utiliza frecuentemente como *backbone* en redes de segmentación semántica ligeras (por ejemplo, Mobile DeepLabV3) y en contextos con restricciones de recursos computacionales, como aplicaciones móviles o drones, aportando buen desempeño sin penalizar la eficiencia energética.

Capítulo 3

Desarrollo

3.1 Dataset INRIA y preprocesado

Para parte del desarrollo de este proyecto se empleó el **INRIA Aerial Image Labeling Dataset** [24], un referente en segmentación semántica urbana sobre imágenes aéreas de alta resolución. Este dataset establece un desafío relevante al plantear una división por ciudades entre los conjuntos de entrenamiento y prueba (train/test), lo que evalúa la capacidad de generalización de los modelos.

Descripción general

- **Número de imágenes**: el conjunto consta de 360 imágenes RGB a color, cada una de tamaño 5000×5000 píxeles, representando 10 ciudades distintas en Estados Unidos y Europa.
- **Resolución espacial**: cada imagen tiene una resolución de 30cm/píxel, lo que permite una segmentación de alta precisión a nivel de edificio.
- **Superficie total**: el dataset cubre aproximadamente 810km^2, distribuidos en 405km^2 para entrenamiento y otros 405km^2 en el conjunto de prueba.
- **Máscaras**: se proporcionan máscaras binarias que delimitan áreas edificadas vs no edificadas, disponibles solo para el conjunto de entrenamiento. El divisionado por ciudades (por ejemplo, Chicago solo en train, San Francisco solo en test) simula la generalización real a nuevos entornos urbanos.

Ventajas y relevancia

Esta base de datos es especialmente exigente debido a:

- Su alta resolución espacial, que exige precisión en los bordes y detalles arquitectónicos.
- El enfoque de separación geográfica, que evalúa la robustez frente a condiciones de iluminación, estilo arquitectónico y patrones urbanos diversos.
- La diversidad de entornos urbanos (distritos densos, suburbios, centros históricos, áreas rurales), lo que brinda representatividad de muchos contextos reales.

En resumen, el dataset INRIA constituye un escenario robusto para entrenar y evaluar modelos de segmentación semántica destinados a la detección de edificaciones, y su estructura permite validar la capacidad de adaptación a nuevas zonas geográficas.

(a) Imagen RGB original

(b) Máscara binaria correspondiente

Figura 3.1: Ejemplo de par imagen-máscara del dataset INRIA.

El lenguaje de programación utilizado en todo el proyecto ha sido **Python** [39].

3.1.1. División y preparación de datos

Para este trabajo se utilizó exclusivamente el conjunto de entrenamiento del **INRIA Aerial Image Labeling Dataset** [24], compuesto por **180 imágenes aéreas a color** de alta resolución (5000 × 5000 píxeles), cada una acompañada por su correspondiente **máscara binaria de segmentación**, donde se etiquetan píxeles como *edificio* o *fondo*. Estas imágenes provienen de cinco ciudades estadounidenses (Austin, Chicago, Kitsap County, Western Tyrol, Vienna), y representan entornos urbanos variados en densidad y morfología.

Dado el tamaño considerable de las imágenes originales (5000 × 5000 píxeles), fue necesario realizar un **preprocesado en dos etapas** para hacer viable su uso en GPU durante el entrenamiento:

- **Primera etapa: división en bloques de 1024×1024 píxeles con solapamiento:** cada imagen y su máscara correspondiente fueron segmentadas en **bloques de** 1024 × 1024 píxeles mediante una ventana deslizante con **30 píxeles de solapamiento** en ambas direcciones, garantizando la cobertura completa y preservando la continuidad espacial en los bordes. Este paso generó un total de **4500 imágenes** con sus máscaras correspondientes.

Figura 3.2: Visualización de los cortes realizados sobre una imagen del dataset INRIA. Los recuadros rojos representan los bloques de 1024x1024 píxeles con un solapamiento de 30 píxeles.

- **Segunda etapa: subdivisión en sub-bloques de 512×512 píxeles:** para aumentar el número de muestras y permitir un preentrenamiento más eficiente, cada bloque de 1024 × 1024 se subdividió sistemáticamente en **cuatro subimágenes de** 512 × 512 píxeles, correspondientes a las regiones superior izquierda, superior derecha, inferior izquierda e inferior derecha. Como resultado, se obtuvieron **18000 imágenes** con sus máscaras correspondientes, todas alineadas espacialmente.

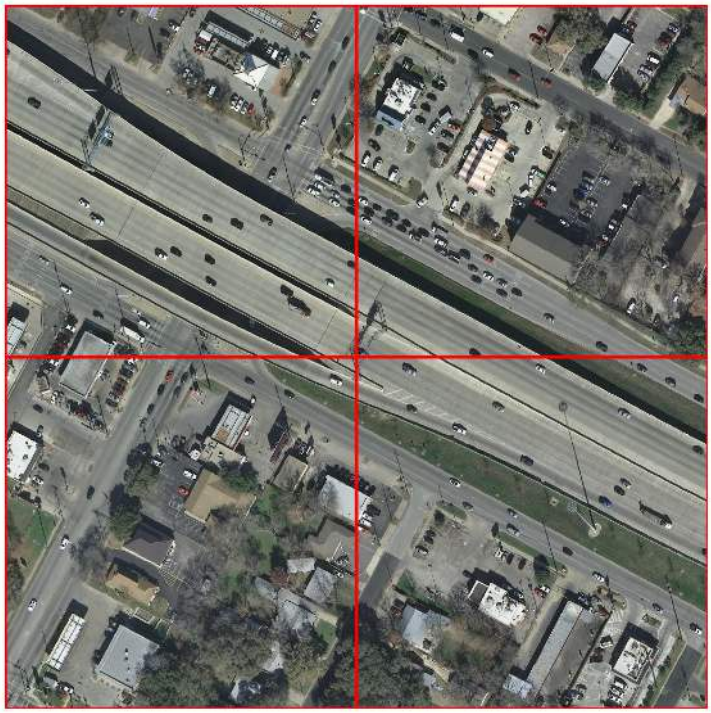


Figura 3.3: Ejemplo visual de subdivisión del bloque de 1024×1024 píxeles en cuatro bloques de 512×512 píxeles.

Este proceso fue implementado mediante **scripts personalizados en Python**, empleando la librería `Pillow` [5] para las operaciones de lectura, recorte y guardado de imágenes. Se aplicaron mecanismos de validación para asegurar la correcta correspondencia entre cada imagen RGB y su máscara binaria, preservando el alineamiento de píxeles durante la segmentación.

Finalmente, el conjunto total de imágenes obtenidas fue dividido en **80 % para entrenamiento** y **20 % para validación**, lo que da un total de 14400 imágenes de entrenamiento y 3600 de validación, garantizando una distribución aleatoria pero consistente entre ambos conjuntos.

3.1.2. Preprocesamiento y transformaciones

Para la etapa de entrenamiento, se definió una clase personalizada `INRIA_Dataset` derivada de `torch.utils.data.Dataset`, lo que permitió cargar las imágenes de manera eficiente junto con sus máscaras correspondientes. Durante esta etapa se aplicaron transformaciones y aumentos de datos mediante la librería **Albumentations** [1], conocida por su rendimiento y flexibilidad en visión por computador.

Para el conjunto de entrenamiento, se aplicaron las siguientes transformaciones:

- **Giro horizontal aleatorio** (`HorizontalFlip`) con probabilidad 0.5.
- **Giro vertical aleatorio** (`VerticalFlip`) con probabilidad 0.5.

- **Rotaciones aleatorias de 90 grados** (`RandomRotate90`) con probabilidad 0.5.
- **Normalización** con la media y desviación estándar de ImageNet: `mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)` [8].
- **Conversión a tensores** mediante `ToTensorV2()` para compatibilidad con PyTorch.

Estas transformaciones tienen como objetivo incrementar la diversidad del conjunto de entrenamiento, mejorar la generalización del modelo y reducir el riesgo de sobreajuste.

Para el conjunto de validación, se utilizaron únicamente las transformaciones deterministas necesarias:

- Normalización con los mismos parámetros de ImageNet.
- Conversión a tensores.

3.2 Creación del dataset PNOA

El **Plan Nacional de Ortofotografía Aérea (PNOA)** es una iniciativa conjunta entre la Administración General del Estado y las Comunidades Autónomas, coordinada por el Instituto Geográfico Nacional (IGN) y el Centro Nacional de Información Geográfica (CNIG). Inserto en el marco del Plan Nacional de Observación del Territorio (PNOT), el PNOA se inició en 2004 con el objetivo de generar ortofotografías digitales homogéneas y de alta resolución en todo el territorio español. Actualmente, se realiza con una periodicidad aproximada de tres años, logrando una resolución espacial de entre 15 y 25cm/píxel en las zonas más actualizadas.

Estas imágenes aéreas se elaboran mediante vuelos fotogramétricos, aerotriangulación, apoyo de campo y generación de modelos digitales de elevación, lo que permite obtener productos georreferenciados de alta precisión, ampliamente utilizados en cartografía, urbanismo, teledetección y seguimiento territorial.

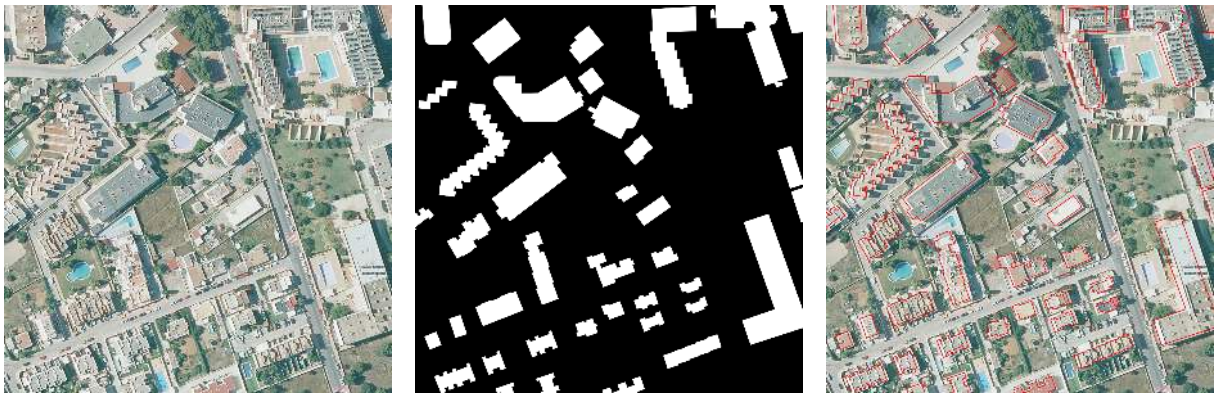
Con el fin de adaptar los modelos preentrenados en el dataset INRIA al entorno geográfico español, se creó un nuevo conjunto de datos a partir de imágenes del PNOA. La generación de las máscaras de edificios se basó en datos extraídos de **OpenStreetMap (OSM)** [6], una base cartográfica colaborativa global, libre y editable, mantenida por una comunidad de más de 10 millones de usuarios voluntarios. Si bien OSM ofrece una cobertura amplia y útil para muchas regiones urbanas, su naturaleza crowdsourced puede implicar desfases temporales o errores en la posición o forma de los elementos, por lo que las máscaras iniciales derivadas de esta fuente requieren una cuidadosa revisión y corrección manual.

De este modo, se procedió a una fase intensiva de limpieza y validación manual sobre las imágenes y sus máscaras, corrigiendo errores de alineación, omisiones y deformaciones, para asegurar que el nuevo dataset resultante reflejara con fidelidad la realidad urbana en el contexto del PNOA.

3.2.1. Visualización y validación de máscaras

Durante esta fase se identificaron numerosos casos en los que las máscaras generadas a partir de los datos vectoriales de OpenStreetMap (OSM) no coincidían con precisión sobre las ortofotografías del PNOA. Estas discrepancias se debieron principalmente a errores de georreferenciación, desfases temporales entre ambas fuentes o a una cobertura incompleta del trazado de edificios en OSM. Tales inconsistencias podían manifestarse como desplazamientos, deformaciones o incluso ausencias totales de edificios en las máscaras, comprometiendo la utilidad de dichos datos para entrenamiento supervisado.

Para facilitar la revisión manual de las muestras, se generaron automáticamente tres versiones por cada imagen: (1) la imagen RGB original del PNOA; (2) su correspondiente máscara binaria de edificios; y (3) una imagen compuesta que superpone sobre la imagen aérea los contornos de los polígonos de las máscaras en color rojo. Esta última versión fue especialmente útil durante el proceso de validación, ya que permitía verificar visualmente, de forma inmediata y clara, el grado de alineación entre las estructuras reales y las regiones etiquetadas.



(a) Imagen RGB original (b) Máscara binaria (c) Máscara superpuesta

Figura 3.4: Comparativa entre la imagen aérea, su máscara y la validación visual con el perímetro de los edificios superpuesto en rojo.

La validación fue llevada a cabo manualmente, inspeccionando individualmente cada muestra. Se descartaron aquellas imágenes con desajustes notables, errores de forma en las máscaras o ausencia significativa de edificaciones relevantes. De un total aproximado de 5000 imágenes de 1024x1024 píxeles revisadas, se conservaron unas 2500 imágenes consideradas como válidas y de calidad suficiente para su uso en la fase de entrenamiento del modelo.

3.2.2. Preparación y normalización de datos

Una vez validado el conjunto de imágenes procedente del PNOA, se aplicó un proceso de preprocesamiento similar al utilizado con el dataset INRIA. Cada imagen validada, de tamaño 1024×1024 píxeles, fue subdividida en cuatro bloques de 512×512 píxeles, con el fin de aumentar el número de muestras disponibles, facilitar un entrenamiento más eficiente en GPU y mejorar la generalización del modelo.

Este procedimiento se llevó a cabo mediante un script personalizado en Python que recorre las imágenes RGB y sus respectivas máscaras binarias, realiza los cortes en las cuatro secciones mencionadas y guarda los nuevos pares imagen-máscara en carpetas organizadas por tipo de dato (imagen o máscara). Como resultado de esta división, se obtuvieron un total aproximado de **10.000 muestras** de tamaño 512×512 píxeles.

Posteriormente, se dividió el conjunto de datos en tres subconjuntos: **70 %** para entrenamiento, **15 %** para validación y **15 %** para test, asegurando una distribución aleatoria pero estratificada para mantener la diversidad de escenas en cada partición.

Durante la fase de entrenamiento, se aplicaron exactamente las mismas transformaciones y aumentos que en el caso del dataset INRIA, empleando la librería **Albumentations** [1].

Para los subconjuntos de validación y test, únicamente se aplicó la normalización y conversión a tensores, garantizando así la coherencia con el flujo de entrenamiento pero sin introducir variaciones aleatorias.

3.3 Entrenamiento de modelos

3.3.1. Configuración y hardware utilizado

El entrenamiento de los modelos se llevó a cabo en un **equipo remoto de altas prestaciones**, configurado desde cero. Este entorno contaba con una **GPU NVIDIA RTX 3090 de 24 GB de VRAM**, cuyo alto rendimiento permitió manejar lotes de imágenes grandes (batch size = 8) y entrenar modelos de segmentación semántica complejos con eficiencia.

La instalación del entorno comenzó con la configuración de **Anaconda** como gestor de entornos y dependencias [33]. Desde Anaconda, se creó un entorno específico de `Python 3.10`, en el cual se instalaron las librerías necesarias para el desarrollo y entrenamiento de modelos. Estas incluyeron:

- `PyTorch` como framework principal de aprendizaje profundo.
- `PyTorch Lightning` [43], que permitió estructurar el código de forma modular, clara y escalable.
- `segmentation_models.pytorch` para acceder a las arquitecturas deseadas.
- `Albumentations` [1] para la aplicación de transformaciones y aumentos de datos.
- `OpenCV` y `Pillow` para la manipulación de imágenes.
- `torchmetrics` para el cálculo de métricas como IoU (Jaccard Index) y F1 Score.
- `Matplotlib` y `JSON` para la visualización y almacenamiento de métricas.

El trabajo se llevó a cabo a través de **notebooks Jupyter** [30], lo cual facilitó una experimentación iterativa y visual del flujo de entrenamiento, validación y análisis de resultados.

El entrenamiento se estructuró usando una clase personalizada llamada `SegmentationModel`, basada en `LightningModule`, lo que permitió organizar todas las fases clave: *forward pass*, *loss function*, *optimización*, *cálculo de métricas* y *registro de resultados*.

Se utilizó como optimizador `Adam` con una tasa de aprendizaje inicial de `1e-3`. Además, se implementó una estrategia de reducción dinámica del *learning rate* usando `ReduceLROnPlateau`, que disminuye el valor del aprendizaje cuando el rendimiento (en términos de IoU de validación) se estanca durante varias épocas consecutivas. El límite se marcó en 3 épocas, y si el rendimiento no mejoraba en esas épocas, se disminuía el learning rate a la mitad.

El número máximo de épocas se fijó en **100**, aunque se integró la técnica de **Early Stopping**, que detiene el entrenamiento automáticamente si no se observa mejora en la métrica de validación tras **5 épocas consecutivas**. Esto evita el sobreajuste y reduce el tiempo de cómputo innecesario.

A su vez, se utilizó **Model Checkpointing** para guardar automáticamente el modelo con mejor rendimiento (mayor `IoU` en validación), permitiendo retomar el entrenamiento o reutilizar el mejor modelo en inferencia posterior.

Durante el entrenamiento se monitorizaron las siguientes métricas:

- **IoU (Jaccard Index)**: principal métrica de calidad en segmentación semántica binaria.
- **F1 Score**: como medida adicional que combina precisión y exhaustividad.
- **Pérdida total (loss)**: combinación de Dice Loss y Binary Cross Entropy.

Todos estos valores fueron registrados por época, tanto en entrenamiento como en validación, y almacenados en archivos `.npy` y `.json`. Además, al final del entrenamiento se generaron visualizaciones automáticas de las curvas de pérdida, IoU y F1 Score mediante `Matplotlib`, lo que permitió evaluar de manera objetiva el progreso y estabilidad del modelo.

Los conjuntos de datos fueron cargados mediante una clase personalizada `INRIA_Dataset`, compatible con `torch.utils.data.Dataset`, y se utilizaron `DataLoaders` con tamaño de lote (*batch size*) $= 8$, sin paralelismo (`num_workers` $= 0$) por compatibilidad con el entorno remoto. Se garantizó el correcto emparejamiento entre imágenes y máscaras y se aplicaron transformaciones distintas para los conjuntos de entrenamiento y validación (como se detalla en secciones anteriores).

En resumen, el sistema de entrenamiento se diseñó para ser reproducible, escalable y eficiente, empleando buenas prácticas de ingeniería de software en el marco de aprendizaje profundo, con registro completo de resultados y configuración flexible para el análisis posterior.

3.3.2. Métricas: IoU, F1-score, Dice, BCE

Durante el entrenamiento y la validación de los modelos se emplearon diversas métricas para evaluar el rendimiento en tareas de segmentación binaria. A continuación, se describen las principales métricas utilizadas, junto con sus respectivas fórmulas:

IoU (Intersection over Union): también conocida como *Índice de Jaccard*, mide el grado de superposición entre la máscara predicha y la máscara real (verdad de terreno). Se define como el cociente entre el área de intersección y el área de la unión de ambas máscaras:

$$\text{IoU} = \frac{TP}{TP + FP + FN}$$

donde TP son los píxeles correctamente predichos como positivos, FP los falsos positivos y FN los falsos negativos. El valor de IoU varía entre 0 (sin superposición) y 1 (superposición perfecta), siendo ampliamente utilizada en segmentación semántica por su equilibrio entre penalización de errores y sensibilidad espacial [31].

F1-score (Coeficiente F1): es la media armónica entre la precisión (*precision*) y el *recall*, representando un balance entre la exactitud de las predicciones positivas y la cobertura de los positivos reales. Su fórmula es:

$$\text{F1} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

Esta métrica es especialmente útil cuando existe cierto desbalance entre clases, como ocurre en segmentación binaria de objetos pequeños frente a grandes áreas de fondo.

Dice Coefficient: también conocido como *Dice Similarity Coefficient (DSC)*, mide la similitud entre dos conjuntos de datos (máscaras en este caso) con énfasis en los verdaderos positivos. Su fórmula es muy similar a la del F1-score, y de hecho son equivalentes en la práctica para clasificación binaria:

$$\text{Dice} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

Aunque comparte la misma expresión que el F1, el Dice se popularizó en el ámbito biomédico y de segmentación estructural. En nuestro caso, se empleó la **Dice Loss**, que simplemente se define como $1 - DiceCoefficient$ para propósitos de minimización durante el entrenamiento [26].

Binary Cross Entropy (BCE): es una función de pérdida estándar para tareas de clasificación binaria. Para cada píxel, compara la probabilidad predicha con la etiqueta real. La fórmula es:

$$\text{BCE} = -\left[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})\right]$$

donde y es la etiqueta binaria real y $\hat{y}$ es la probabilidad predicha. Esta pérdida penaliza con mayor intensidad los errores de clasificación más seguros (cuando se predice con mucha confianza una clase incorrecta).

Función de pérdida combinada: en el entrenamiento se utilizó una función de pérdida compuesta que combina de forma aditiva la **Dice Loss** y la **Binary Cross Entropy (BCE)**:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{Dice}} + \mathcal{L}_{\text{BCE}}$$

Esta combinación permite aprovechar lo mejor de ambas: la BCE actúa a nivel local penalizando errores píxel a píxel, mientras que la Dice Loss evalúa la similitud general entre las máscaras. Esta sinergia mejora la estabilidad y la precisión global de los modelos [15].

3.4 Evaluación preliminar con INRIA

En esta etapa se llevó a cabo una evaluación preliminar de los modelos utilizando exclusivamente el conjunto de datos INRIA. Se consideraron cuatro modelos: UPerNet-ConvNext_base, DeepLabV3+-ResNeXt101_32x8d, UNet-ResNet34 y PAN-MobileNet_v2.

Se analizaron cuidadosamente las curvas de entrenamiento y validación obtenidas durante el proceso. En particular, se estudiaron la evolución de la función de pérdida (*loss*), el índice de intersección sobre unión (*IoU*) y el coeficiente F1 (*F1-score*), permitiendo evaluar el aprendizaje y la convergencia de cada modelo a lo largo de las épocas de entrenamiento.

Análisis de curvas del modelo UNet-ResNet34

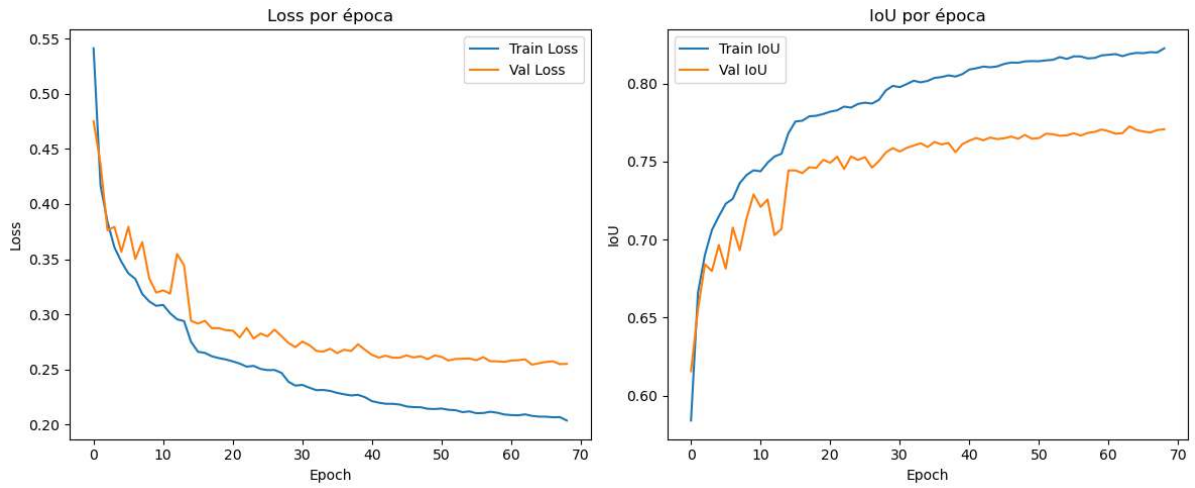


Figura 3.5: Curvas de entrenamiento y validación del modelo **UNet-ResNet34** para la función de pérdida (*Loss*) y el índice de intersección sobre unión (*IoU*).

En las gráficas del modelo **UNet-ResNet34** se observa una reducción estable y progresiva en la pérdida (*loss*), tanto en entrenamiento como en validación. A partir de la época 30 aproximadamente, se aprecia una estabilización clara de la pérdida de validación, lo que indica una convergencia efectiva sin signos evidentes de sobreajuste.

En cuanto al índice IoU, ambas curvas muestran una clara tendencia ascendente, con una mejora continua hasta estabilizarse alrededor de la época 50. La separación entre ambas curvas sugiere un ligero sobreajuste, aunque el modelo mantiene un desempeño sólido en datos no vistos.

Cabe destacar que, debido a limitaciones técnicas durante el entrenamiento inicial, no se dispone de la curva del F1-score para este modelo, considerándose las métricas de pérdida e IoU suficientes para la evaluación preliminar.

Análisis de curvas del modelo DeepLabV3+-ResNeXt101_32x8d

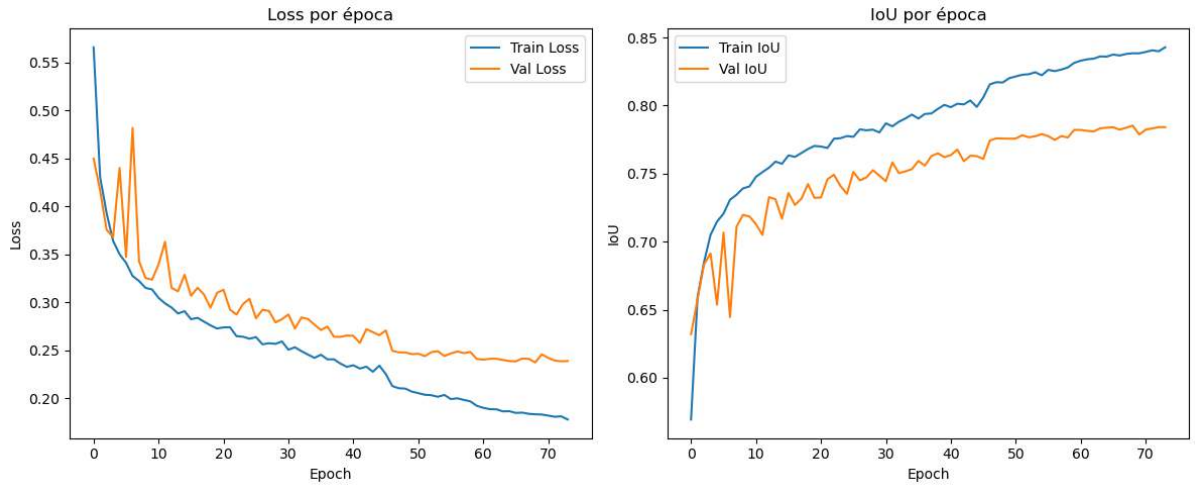


Figura 3.6: Curvas de entrenamiento y validación del modelo **DeepLabV3+-ResNeXt101_32x8d** para la función de pérdida (*Loss*) y el índice de intersección sobre unión (*IoU*).

Las curvas del modelo **DeepLabV3+-ResNeXt101_32x8d** muestran un comportamiento general positivo. La pérdida en entrenamiento y validación desciende continuamente durante las primeras 30 épocas, estabilizándose gradualmente hacia épocas posteriores. Aunque se observa cierta separación entre las curvas de entrenamiento y validación, lo que indica un leve sobreajuste, el modelo sigue manteniendo un rendimiento estable en validación a lo largo del entrenamiento.

En el índice IoU, ambas curvas presentan un crecimiento estable, mostrando una rápida convergencia inicial y alcanzando estabilidad alrededor de la época 40. La ligera distancia persistente entre las curvas de entrenamiento y validación refuerza el leve sobreajuste observado en la pérdida, pero sin comprometer significativamente la generalización del modelo.

Debido a limitaciones técnicas, no se dispone de la curva del *F1-score* para este modelo, siendo la pérdida e IoU suficientes para una evaluación preliminar adecuada.

Análisis de curvas del modelo UPerNet-ConvNext_base

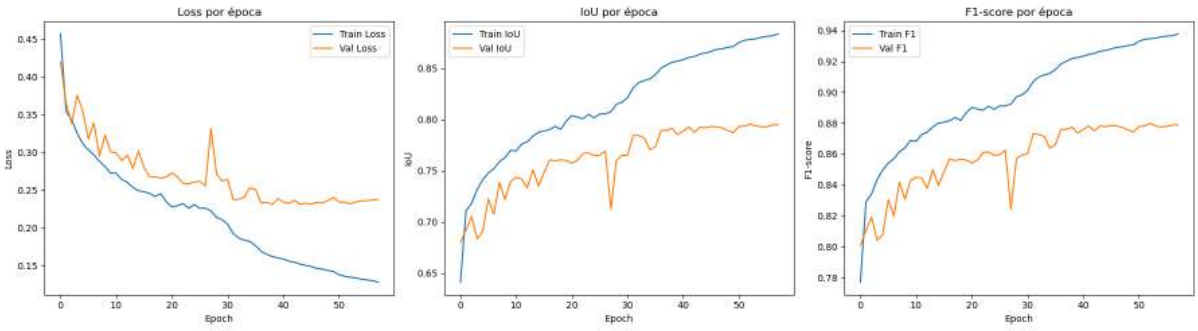


Figura 3.7: Curvas de entrenamiento y validación del modelo **UPerNet-ConvNext_base** para la función de pérdida (*Loss*), índice de intersección sobre unión (*IoU*) y coeficiente F1 (*F1-score*).

El modelo **UPerNet-ConvNext_base** muestra un comportamiento altamente positivo en sus curvas de entrenamiento y validación. La curva de pérdida (*Loss*) muestra una reducción progresiva y estable hasta aproximadamente la época 30, donde empieza a estabilizarse manteniendo buenos valores en ambas curvas, lo que indica una adecuada convergencia sin un sobreajuste pronunciado.

La curva del índice IoU refleja claramente un incremento rápido y sostenido durante las primeras épocas, estabilizándose alrededor de la época 35 con valores altos tanto en entrenamiento como en validación, sugiriendo un rendimiento robusto del modelo.

Además, la curva del *F1-score* presenta un patrón similar al IoU, aumentando rápidamente y estabilizándose posteriormente en valores elevados. Esta métrica adicional refuerza la conclusión de que el modelo presenta una excelente capacidad para identificar correctamente las áreas de interés.

En conjunto, las métricas indican un rendimiento sólido, destacando la eficacia del modelo para segmentación semántica con buena generalización sobre datos no vistos.

Análisis de curvas del modelo PAN-MobileNet_v2

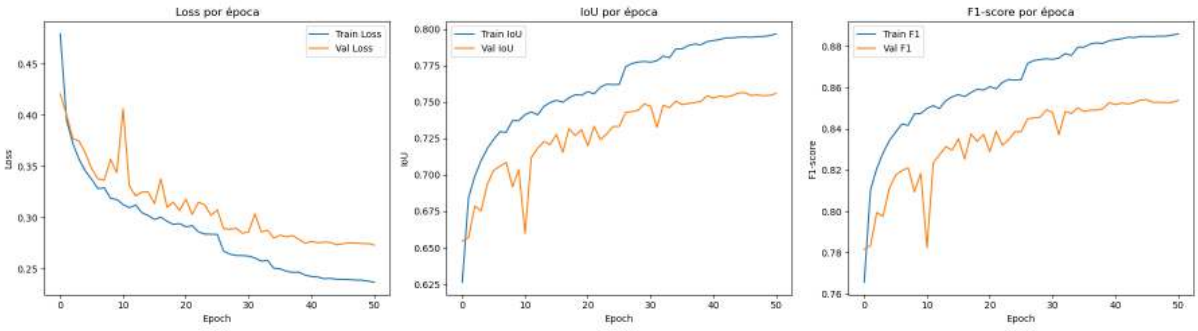


Figura 3.8: Curvas de entrenamiento y validación del modelo **PAN-MobileNet_v2** para la función de pérdida (*Loss*), índice de intersección sobre unión (*IoU*) y coeficiente F1 (*F1-score*).

Las curvas obtenidas del modelo **PAN-MobileNet_v2** revelan un buen comportamiento general del entrenamiento. En la curva de pérdida, se observa una reducción rápida inicial seguida por una estabilización gradual a partir de la época 25 aproximadamente, con una separación moderada entre las curvas de entrenamiento y validación, indicando un ligero sobreajuste controlado.

En cuanto al índice IoU, ambas curvas muestran un crecimiento estable hasta estabilizarse alrededor de la época 30, confirmando un rendimiento consistente del modelo en términos de segmentación semántica.

El comportamiento del *F1-score* es coherente con la evolución del IoU, alcanzando valores elevados tras una rápida mejora inicial. Esto sugiere una sólida capacidad predictiva del modelo en la identificación de regiones correctas, complementando la información aportada por la pérdida e IoU.

En general, el modelo PAN-MobileNet_v2 muestra una capacidad de generalización aceptable y buen equilibrio entre precisión y capacidad predictiva, adecuado para aplicaciones donde la eficiencia computacional es una prioridad.

Para ilustrar visualmente el rendimiento de los modelos, se generaron predicciones sobre ejemplos representativos del conjunto de validación. En cada caso se presentan las siguientes imágenes comparativas:

- Imagen RGB original.
- *Ground truth*: Máscara real.
- Predicción generada por el modelo.
- *Overlay* visual, donde el color verde indica el ground truth, el rojo la predicción del modelo y el amarillo la intersección entre ambos.

- Mapa de error visual, con la siguiente leyenda: Falsos Negativos (FN) en azul, Falsos Positivos (FP) en rojo y Verdaderos Positivos (TP) en verde.

Estos ejemplos permiten identificar de manera visual las fortalezas y debilidades específicas de cada modelo.

Análisis visual de predicciones del modelo UNet-ResNet34

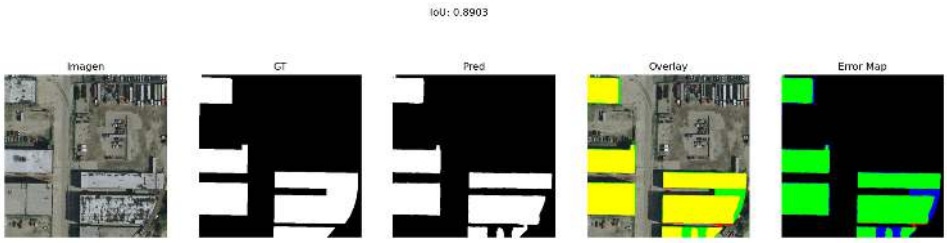


Figura 3.9: Ejemplo visual del modelo **UNet-ResNet34**. De izquierda a derecha: Imagen RGB, máscara real (GT), predicción del modelo, *overlay* (GT verde, predicción roja, intersección amarilla) y mapa de error (TP verde, FP rojo, FN azul). IoU obtenido: 0.8903.

En este ejemplo se presentan resultados visuales del modelo **UNet-ResNet34**, mostrando la imagen RGB original, la máscara de verdad de terreno (*ground truth*), la predicción realizada por el modelo, un *overlay* para comparar visualmente ambas máscaras, y un mapa de error donde se identifican los verdaderos positivos (verde), falsos positivos (rojo) y falsos negativos (azul).

Fortalezas:

- El modelo identifica eficazmente la mayoría de las edificaciones presentes en la imagen, con un valor de IoU bastante elevado (0.8903), indicando una alta precisión general.
- Las áreas principales y más grandes están correctamente segmentadas, observándose una correspondencia casi perfecta entre la predicción y la máscara real (áreas amarillas en el *overlay*, verdes en el mapa de error).

Debilidades:

- Existen ciertas dificultades en bordes y estructuras más pequeñas o estrechas, donde se generan falsos negativos (en azul) indicando que algunos edificios o partes de estos no son correctamente detectados.
- También aparecen pequeñas áreas en rojo en el mapa de error, correspondientes a falsos positivos, donde el modelo predice presencia de edificación en áreas donde

realmente no existen. Esto suele ocurrir en zonas de transición o sombras proyectadas.

En resumen, aunque el modelo UNet-ResNet34 demuestra un sólido desempeño general, es susceptible a errores menores en la segmentación precisa de bordes y áreas reducidas, aspectos que podrían mejorar con técnicas adicionales de refinamiento o ajustes en el entrenamiento.

Análisis visual de predicciones del modelo DeepLabV3+-ResNeXt101_32x8d

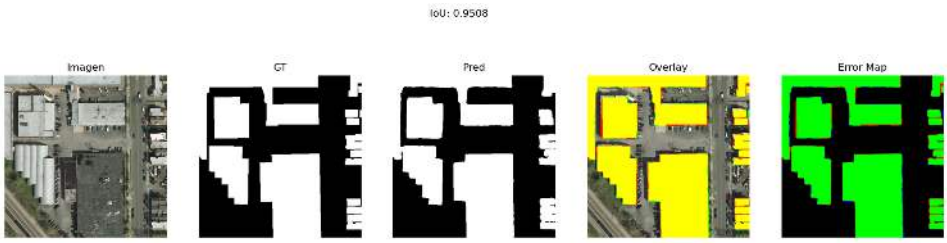


Figura 3.10: Ejemplo visual del modelo **DeepLabV3+-ResNeXt101_32x8d**. De izquierda a derecha: Imagen RGB, máscara real (GT), predicción del modelo, *overlay* (GT verde, predicción roja, intersección amarilla) y mapa de error (TP verde, FP rojo, FN azul). IoU obtenido: 0.9508.

En la Figura 3.10 se muestran resultados visuales obtenidos por el modelo **DeepLabV3+-ResNeXt101_32x8d**. El modelo alcanza un excelente rendimiento con un **IoU de 0.9508**, lo que refleja una segmentación muy precisa.

Fortalezas:

- La segmentación de edificios es notablemente precisa, tanto en grandes estructuras como en edificaciones pequeñas o con formas irregulares.
- Se observa una fuerte coincidencia entre la predicción y la verdad de terreno, reflejada en la gran cantidad de regiones amarillas en el *overlay* y verdes en el mapa de error.
- Las transiciones entre bordes están bien ajustadas, lo que sugiere una alta capacidad del modelo para aprender contornos complejos.

Debilidades:

- Aunque mínimas, se detectan ligeras áreas con falsos positivos (en rojo) en bordes exteriores, lo que puede deberse a sombras u otros elementos urbanos que el modelo confunde con edificaciones.
- También se aprecian falsos negativos puntuales (azul), especialmente en zonas estrechas o líneas delgadas de edificaciones que podrían ser más difíciles de identificar con precisión.

En general, este modelo destaca por su robustez y alto rendimiento en la tarea de segmentación semántica, mostrando un equilibrio excelente entre precisión y cobertura.

Análisis visual de predicciones del modelo UPerNet-ConvNext_base

Figura 3.11: Ejemplo visual del modelo **UPerNet-ConvNext_base**. De izquierda a derecha: imagen RGB, máscara real (GT), predicción, superposición (*overlay*: GT en verde, predicción en rojo, coincidencias en amarillo) y mapa de error (TP en verde, FP en rojo, FN en azul). IoU obtenido: 0.9089, F1-score: 0.9523.

La Figura 3.11 muestra un ejemplo representativo de las predicciones del modelo **UPerNet-ConvNext_base**, el cual alcanza un **IoU de 0.9089** y un **F1-score de 0.9523**, valores que reflejan un rendimiento sólido y consistente.

Fortalezas:

- Se observa una segmentación muy precisa del edificio principal, con alta coincidencia entre la máscara predicha y la verdad de terreno, especialmente evidente en el *overlay* (zonas amarillas) y en el mapa de error (predominancia de verde).
- La arquitectura es capaz de capturar de forma adecuada formas regulares y bordes rectos, preservando la geometría del objeto.
- La predicción se adapta correctamente incluso en bordes parcialmente ocultados por sombras o árboles.

Debilidades:

- Se presentan ligeros falsos positivos (rojo) en zonas periféricas del entorno urbano, como pequeñas estructuras no edificadas.
- También se detectan algunos falsos negativos (azul) en tejados más pequeños del fondo, lo que puede indicar una ligera sensibilidad reducida a estructuras de menor tamaño o con bajo contraste visual respecto al fondo.

En términos generales, el modelo UPerNet-ConvNext_base demuestra un alto nivel de generalización y robustez, siendo especialmente eficaz en entornos urbanos bien estructurados y con edificaciones predominantes de gran tamaño.

Análisis visual de predicciones del modelo PAN-MobileNet v2

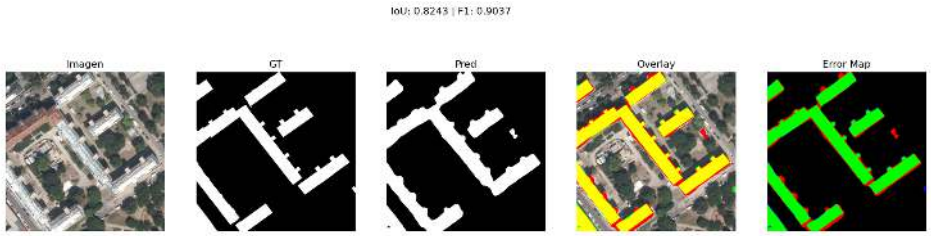


Figura 3.12: Ejemplo de segmentación con PAN-MobileNet_v2 (IoU = 0.8243, F1 = 0.9037). De izquierda a derecha: Imagen original, Ground Truth, Predicción, Superposición (verde: GT, rojo: predicción, amarillo: coincidencias), Mapa de error (verde: TP, rojo: FP, azul: FN).

En este ejemplo, el modelo **PAN-MobileNet_v2** logra un rendimiento global aceptable con un **IoU de 0.8243** y un **F1-score de 0.9037**. Como puede observarse en la figura, el modelo detecta correctamente la mayoría de las edificaciones principales, destacando una buena cobertura sobre los tejados amplios y estructuras rectangulares.

Sin embargo, presenta algunas **debilidades notables** en zonas más complejas o irregulares. En particular, se aprecian errores en los bordes de edificios delgados o adyacentes, donde se generan *falsos negativos* (en azul) y *falsos positivos* (en rojo) más frecuentes que en modelos más complejos. Esto puede deberse a las limitaciones de capacidad del backbone MobileNet_v2, diseñado para entornos con recursos computacionales limitados, lo que compromete cierta precisión.

En conjunto, PAN-MobileNet_v2 representa una alternativa eficiente en cuanto a coste computacional, aunque con un rendimiento algo inferior respecto a modelos más pesados como DeepLabV3+ o UPerNet.

Estas fortalezas y debilidades están directamente relacionadas con las características intrínsecas del dataset INRIA. Este conjunto de datos está compuesto por imágenes aéreas de alta resolución con etiquetas precisas generadas manualmente, lo que favorece que los modelos capten con mayor facilidad estructuras grandes y bien definidas. Sin embargo, las dificultades aparecen en situaciones ambiguas o con detalles finos, probablemente debido a la escala de los modelos y la resolución efectiva del entrenamiento.

Esta evaluación preliminar permitió obtener una primera referencia sólida sobre el desempeño de cada modelo, estableciendo una base para el proceso posterior de *fine-tuning* con datos del PNOA, adaptados específicamente al contexto territorial español.

3.5 Fine-tuning sobre datos del PNOA

Objetivo del Fine-tuning

Con el fin de adaptar los modelos entrenados inicialmente sobre el conjunto de datos **INRIA** al contexto geográfico español, se realizó una fase de **fine-tuning** empleando imágenes y máscaras del **Plan Nacional de Ortofotografía Aérea (PNOA)**. Esta etapa tuvo como propósito mejorar la capacidad de generalización de los modelos frente a características urbanas y arquitectónicas propias del territorio nacional, diferentes a las presentes en el dataset original.

Para abordar esta tarea se aplicó la estrategia de **transfer learning**, que permite aprovechar el conocimiento adquirido en un dominio fuente (en este caso, INRIA) y refinarlo sobre un nuevo dominio objetivo (PNOA). Esta técnica resulta especialmente eficaz cuando el nuevo conjunto de datos es más reducido o presenta características distintas pero relacionadas, evitando así los costes computacionales de entrenar desde cero.

En concreto, se partió de los modelos preentrenados con INRIA y se continuó su entrenamiento sobre los datos del PNOA. En esta etapa se evaluaron dos configuraciones: *congelar parcial o totalmente el encoder* (responsable de extraer características generales) o mantenerlo entrenable. La congelación se aplicó a todos los modelos excepto a **UPerNet-ConvNext_base**, en el que se optó por dejar el encoder completamente ajustable debido a su arquitectura más reciente y su capacidad de adaptación. Esta decisión permitió evaluar si preservar características generales del preentrenamiento resultaba beneficioso para el nuevo dominio o, por el contrario, impedía una adaptación efectiva.

Configuración del Fine-tuning

En la fase de fine-tuning, se mantuvo la misma estructura general empleada en la etapa de preentrenamiento con el conjunto de datos INRIA, pero se introdujeron dos modificaciones clave orientadas a facilitar una adaptación gradual al nuevo dominio.

La primera modificación consistió en la reducción de la **tasa de aprendizaje** a `1e-4`, un valor más conservador que permite realizar ajustes más finos sobre los pesos previamente entrenados, evitando sobrescribir abruptamente el conocimiento aprendido en la fase inicial.

La segunda modificación fue la aplicación, en la mayoría de los modelos, de una **congelación parcial del encoder**. Esta técnica impide que ciertas capas del extractor de características se actualicen durante el entrenamiento, con el objetivo de conservar representaciones generales adquiridas previamente en el dataset INRIA. La profundidad de la congelación dependía de la arquitectura concreta, permitiendo así evaluar el equilibrio entre preservación del conocimiento previo y capacidad de adaptación al nuevo contexto. El único modelo en el que no se aplicó ningún tipo de congelación fue **UPerNet-ConvNext_base**, dada su arquitectura más moderna y flexible.

Esquema de congelación de capas

En el contexto del aprendizaje profundo, **congelar una capa** implica impedir que sus pesos se actualicen durante el proceso de entrenamiento. Técnicamente, esto se logra estableciendo la propiedad `requires_grad=False` en los parámetros de dicha capa, lo que indica a PyTorch que no debe calcular gradientes ni aplicar actualizaciones sobre ellos mediante retropropagación.

Esta estrategia se emplea comúnmente en técnicas de **transfer learning**, permitiendo reutilizar características generales aprendidas previamente por un modelo (por ejemplo, sobre un dataset como ImageNet o INRIA), mientras se ajustan únicamente las capas posteriores para adaptarse a un nuevo dominio de datos — en este caso, las imágenes aéreas del territorio español provenientes del PNOA.

Esquema de congelación en UNet-ResNet34

En la Figura 3.13 se presenta un diagrama esquemático del modelo **UNet con encoder ResNet34**, en el cual se indican las capas del encoder que han sido *congeladas* durante el proceso de *fine-tuning* sobre los datos del PNOA. Congelar una capa implica establecer `requires_grad=False`, lo cual evita que sus pesos sean actualizados durante el entrenamiento, preservando el conocimiento previamente adquirido (en este caso, a partir del preentrenamiento con el conjunto de datos INRIA).

En este modelo, se han congelado las siguientes capas del encoder: `conv1`, `bn1`, `layer1` y `layer2`. Estas capas se muestran en color azul y con el símbolo de un copo de nieve, indicando explícitamente que fueron mantenidas fijas. Las capas restantes (decoder y capas profundas del encoder) son entrenables.

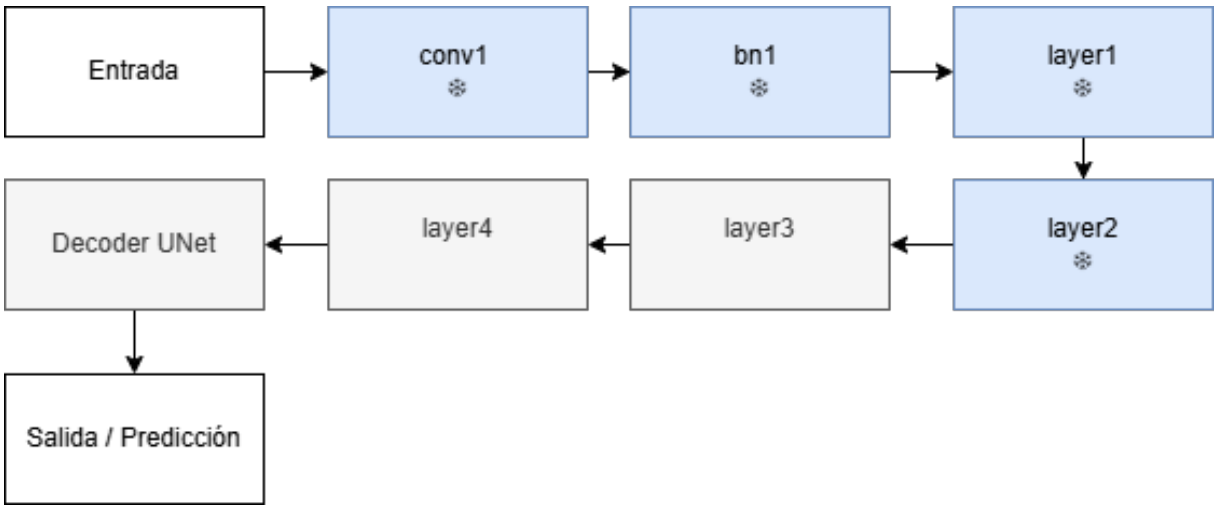


Figura 3.13: Esquema del modelo UNet-ResNet34. Las capas en azul y marcadas con el icono de un copo de nieve representan módulos congelados (`requires_grad=False`) durante el fine-tuning.

Esquema de congelación en DeepLabV3+-ResNeXt101_32x8d

La Figura 3.14 muestra un diagrama esquemático de la arquitectura **DeepLabV3+** con encoder **ResNeXt101_32x8d**, indicando visualmente las capas que han sido conge-

ladas durante el fine-tuning sobre los datos del PNOA.

Al igual que en otros modelos con backbone ResNet/ResNeXt, se han congelado las capas iniciales del encoder: `conv1`, `bn1`, `layer1` y `layer2`, marcadas en azul y con el símbolo de un copo de nieve. Estas capas preservan características visuales genéricas aprendidas previamente con el conjunto INRIA. El resto de la arquitectura, incluyendo capas profundas del encoder, el módulo ASPP y el decoder, permanece entrenable.

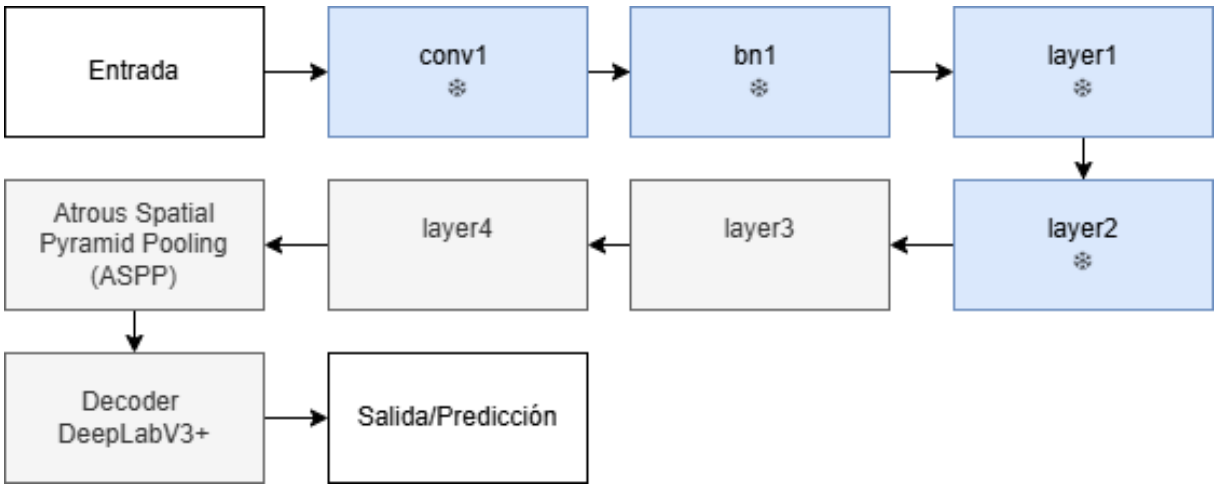


Figura 3.14: Esquema del modelo DeepLabV3+-ResNeXt101_32x8d. Las capas en azul y con el símbolo de un copo de nieve indican que han sido congeladas (`requires_grad=False`) durante el fine-tuning.

Esquema de congelación en PAN-MobileNetV2

La Figura 3.15 representa un esquema simplificado del modelo **PAN-MobileNetV2**, utilizado para el fine-tuning con el conjunto PNOA. En este modelo, se congelaron las primeras capas del encoder correspondiente a `MobileNetV2`, concretamente: `features.0`, `features.1`, `features.2` y `features.3`.

Estas capas, que capturan patrones visuales básicos, se marcaron en color azul y con el icono de un copo de nieve para indicar que no se actualizaron durante el entrenamiento. El resto del encoder, así como los bloques de atención piramidal y el decodificador, permanecieron entrenables.

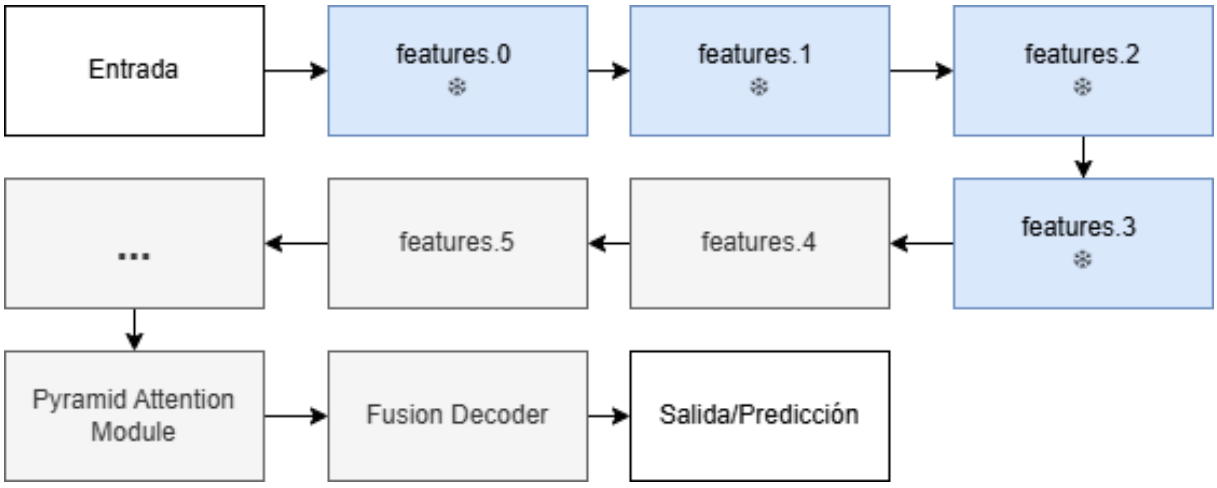


Figura 3.15: Esquema del modelo PAN-MobileNetV2. Las capas azules con el símbolo de un copo de nieve indican que han sido congeladas durante el fine-tuning.

Esquema de congelación en UPerNet-ConvNeXt_base

A diferencia de los modelos anteriores, en el caso de **UPerNet-ConvNeXt_base** no se aplicó congelación de capas durante el proceso de *fine-tuning*. Es decir, todos los bloques de la arquitectura, incluido el encoder basado en `ConvNeXt_base`, fueron actualizados durante el entrenamiento.

Esta decisión se tomó con el objetivo de permitir al modelo adaptarse completamente a las características específicas del conjunto de datos del PNOA, ya que se trata de un backbone más reciente y robusto, con una mayor capacidad de generalización.

Al no haber capas congeladas, no se incluye un esquema visual en esta sección.

Evaluación visual y comparativa con INRIA

UNet-ResNet34

Curvas de entrenamiento.

La Figura 3.16 muestra la evolución de las métricas durante el proceso de fine-tuning del modelo UNet con backbone ResNet34 sobre el conjunto de datos del PNOA. Se observa una reducción progresiva de la función de pérdida tanto en entrenamiento como en validación, acompañada de un incremento en las métricas IoU y F1. No obstante, el ritmo de mejora tiende a estabilizarse pronto, lo que sugiere una capacidad limitada de adaptación al nuevo dominio geográfico.

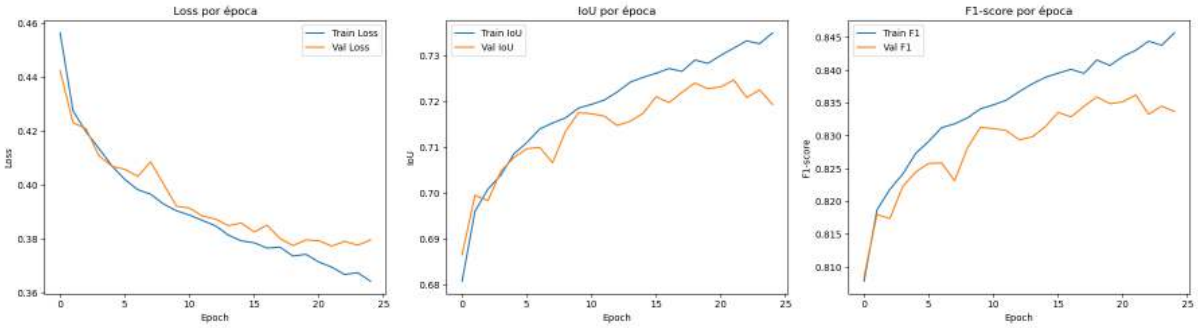


Figura 3.16: Curvas de entrenamiento de UNet-ResNet34 durante el fine-tuning con datos del PNOA.

Congelación del encoder.

En este caso, se congelaron las capas iniciales del encoder (`conv1`, `bn1`, `layer1` y `layer2`) para preservar las representaciones generales aprendidas en INRIA. Esta estrategia busca evitar el sobreajuste y reducir el tiempo de ajuste. Aun así, la mejora sobre los datos del PNOA ha sido limitada, y se evidencia que las características aprendidas no se transfieren de forma totalmente efectiva al nuevo dominio.

Ejemplo visual.

La Figura 3.17 presenta un ejemplo visual del rendimiento del modelo tras el fine-tuning. A pesar de capturar correctamente estructuras urbanas grandes, se observan errores en edificaciones pequeñas o con geometrías complejas. El overlay revela una mayor cantidad de falsos negativos (en rojo), principalmente en los bordes y en zonas densamente edificadas.

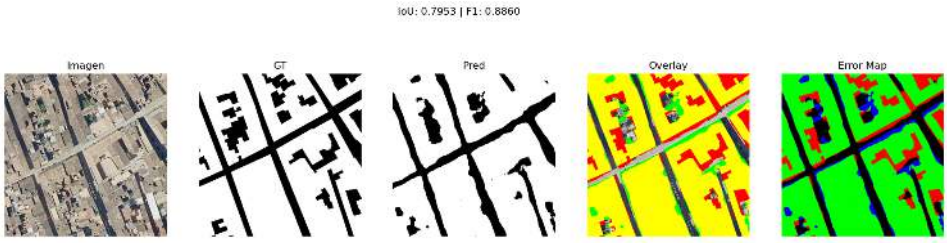


Figura 3.17: Ejemplo visual de predicción del modelo UNet-ResNet34 sobre una imagen del PNOA tras fine-tuning.

Comparativa con INRIA.

Comparando con el desempeño obtenido sobre INRIA, el rendimiento ha empeorado ligeramente tras el fine-tuning. Este descenso indica una menor capacidad de generalización al nuevo dominio, probablemente debido a diferencias notables en el estilo visual de las imágenes o distribución espacial de las edificaciones.

Fortalezas y debilidades.

- **Fortalezas:** Buena detección de estructuras principales. El modelo mantiene una capacidad aceptable de segmentación en regiones bien definidas y de gran tamaño.
- **Debilidades:** Sensibilidad a la variación en texturas, sombras y edificaciones pequeñas. Reducción del rendimiento en bordes y zonas complejas. El fine-tuning con encoder parcialmente congelado no ha sido suficiente para adaptar completamente el modelo al dominio español.

DeepLabV3+-ResNeXt101_32x8d

Curvas de entrenamiento.

La Figura muestra la evolución de la función de pérdida, el IoU y el F1-score durante el proceso de fine-tuning sobre el conjunto del PNOA. Se observa una convergencia estable, con una tendencia positiva tanto en el rendimiento de entrenamiento como en validación, alcanzando valores de F1 cercanos a 0.84 en validación. No se aprecia un sobreajuste marcado.

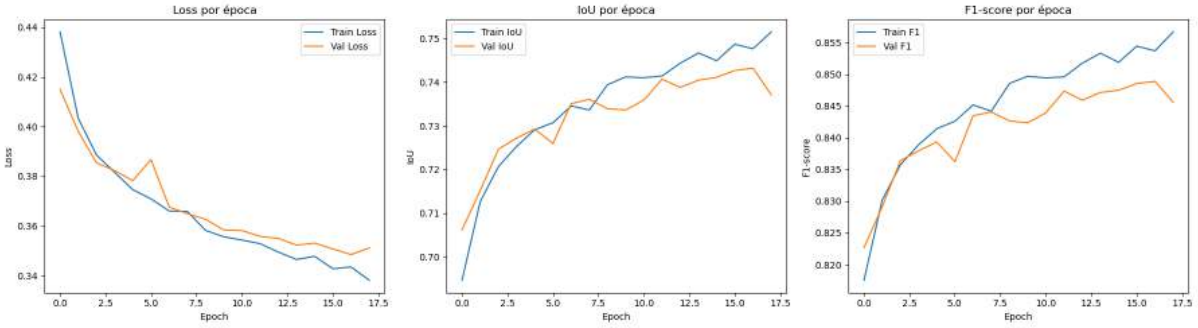


Figura 3.18: Curvas de entrenamiento de DeepLabV3+-ResNeXt101_32x8d durante el fine-tuning con datos del PNOA.

Congelación del encoder.

En este experimento, se aplicó **congelación parcial del encoder**, específicamente sobre las capas iniciales: `conv1`, `bn1`, `layer1` y `layer2`, tal como se detalla en la sección correspondiente al esquema de congelación. Esta estrategia buscaba preservar características generales aprendidas en INRIA, permitiendo adaptar las capas superiores al nuevo dominio PNOA.

Ejemplo visual.

La Figura muestra una imagen RGB del PNOA junto a la máscara de referencia (GT), la predicción del modelo, el solapamiento entre ambas y el mapa de errores:

- El modelo ha logrado detectar con precisión una única edificación en la escena.

- El overlay indica una coincidencia muy alta entre predicción y GT, con bordes ligeramente suavizados.
- El mapa de errores confirma la precisión: casi toda la zona aparece en verde (acierto), con bordes en rojo y azul muy delgados.

Figura 3.19: Ejemplo visual de predicción del modelo DeepLabV3+-ResNeXt101_32x8d sobre una imagen del PNOA tras fine-tuning.

Comparativa con INRIA.

Comparando con los resultados previos sobre INRIA, se observa una **ligera degradación del rendimiento** en el nuevo dominio. No obstante, estas métricas siguen siendo elevadas, lo que sugiere una buena capacidad de generalización.

Fortalezas y debilidades.

- **Fortalezas:** Alta capacidad para generalizar a nuevas imágenes tras fine-tuning, segmentación precisa de edificios, incluso con geometrías limpias, y buen equilibrio entre precisión y robustez espacial.
- **Debilidades:** Ligeramente más conservador en los bordes respecto a INRIA, posiblemente por la menor cantidad de datos del PNOA o su variabilidad espacial, y algunas pérdidas sutiles en los contornos, que podrían corregirse con un ajuste más fino de learning rate o descongelando más capas.

UPerNet-ConvNeXt_base

Durante el proceso de fine-tuning del modelo UPerNet-ConvNeXt_base, se optó por no congelar ninguna de las capas del encoder, permitiendo así una adaptación completa al dominio geográfico del PNOA. Esta decisión se tomó con la intención de maximizar la capacidad de aprendizaje específico del nuevo entorno, ya que esta arquitectura moderna está diseñada para beneficiarse del ajuste completo de sus parámetros.

Curvas de entrenamiento. Las gráficas muestran una convergencia progresiva y coherente del modelo, especialmente en entrenamiento. Sin embargo, se observa cierta divergencia entre las métricas de entrenamiento y validación a medida que avanzan las épocas. Esto podría indicar una tendencia leve al sobreajuste.

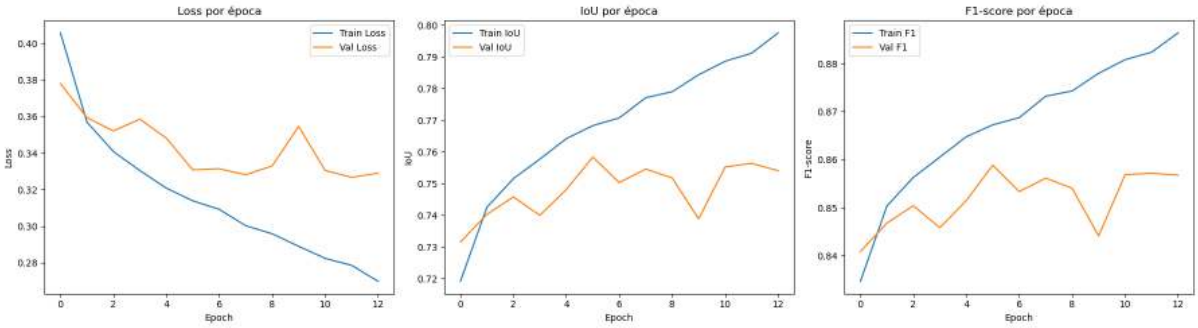


Figura 3.20: Curvas de entrenamiento para UPerNet-ConvNeXt_base sobre PNOA.

Ejemplo visual.

A continuación, se presenta un ejemplo visual del rendimiento del modelo sobre una imagen del conjunto de test del PNOA. Se incluyen la imagen RGB original, la máscara de referencia (GT), la predicción del modelo, la superposición con colores y el mapa de errores.

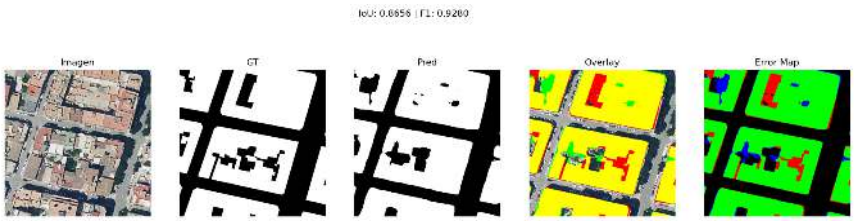


Figura 3.21: Ejemplo de predicción del modelo UPerNet-ConvNeXt_base tras fine-tuning sobre PNOA. IoU: 0.8656 — F1: 0.9280.

Comparativa con INRIA.

En la evaluación previa sobre el dataset INRIA, este modelo logró buenas métricas. Tras el fine-tuning sobre PNOA, sus resultados empeoraron levemente, lo cual sugiere que el modelo tenía un buen desempeño generalizado y que una re-adaptación completa a un nuevo dominio no resultó particularmente beneficiosa en este caso.

Fortalezas y debilidades.

- **Fortalezas:** Excelente preservación de detalles arquitectónicos y precisión en bordes y mínimos errores de omisión o comisión en estructuras bien definidas.
- **Debilidades:** Ligera pérdida de generalización tras el fine-tuning completo y aumento del ruido en regiones con geometría irregular o en sombras.

PAN-MobileNetV2

Curvas de entrenamiento.

En la Figura 3.22, se observan las curvas correspondientes al entrenamiento mediante fine-tuning sobre el dataset PNOA. Las métricas de entrenamiento muestran una mejora continua sin signos de sobreajuste notable: el F1 de validación alcanza valores cercanos al de entrenamiento y el loss se mantiene decreciente.

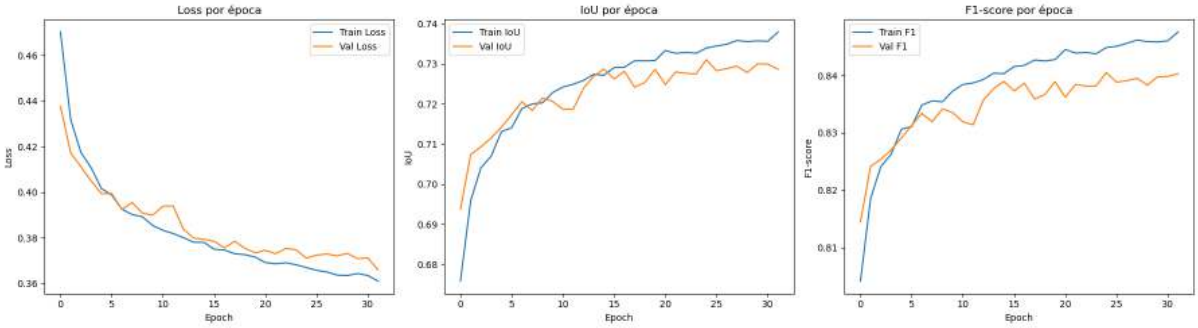


Figura 3.22: Curvas de entrenamiento del modelo PAN-MobileNetV2 durante el fine-tuning sobre PNOA.

Congelación del encoder.

Para este modelo se congelaron las capas iniciales del encoder `mobilenet_v2`, concretamente los bloques `features.0` a `features.3`, correspondientes a las primeras convoluciones y capas de bajo nivel. Esta estrategia permitió preservar características generales aprendidas en INRIA mientras se adaptaban capas más profundas a las particularidades del PNOA.

Ejemplo visual.

La Figura 3.23 muestra un ejemplo representativo del modelo sobre una imagen de prueba del PNOA. Se incluye la imagen RGB original, la verdad de terreno (GT), la predicción binaria generada, una superposición (overlay) y un mapa de errores.

Figura 3.23: Ejemplo visual del modelo PAN-MobileNetV2 sobre PNOA. IoU: 0.7853 — F1: 0.8797.

Comparativa con INRIA.

Comparando con el rendimiento previo en INRIA, el modelo muestra un leve descenso en ambas métricas al aplicarse sobre el PNOA, a pesar del proceso de fine-tuning. Esto sugiere que la adaptación al dominio español no fue completamente efectiva, posiblemente por la capacidad limitada del encoder ligero `mobilenet_v2` para aprender nuevas representaciones con pocas capas entrenables.

Fortalezas y debilidades.

- **Fortalezas:** Buena generalización inicial gracias al encoder preentrenado, tamaño compacto y eficiente para despliegue en entornos con recursos limitados y el overlay y mapa de errores revelan buena delineación de los edificios, especialmente en zonas limpias y de contraste alto.
- **Debilidades:** Menor capacidad de adaptación a texturas y geometrías nuevas respecto a modelos más profundos, rendimiento inferior al de modelos como DeepLabV3+ o UPerNet y sensibilidad a la resolución y estilo visual del nuevo dominio.

Discusión general

El proceso de fine-tuning sobre el dataset PNOA no supuso una mejora sistemática del rendimiento respecto al entrenamiento original sobre INRIA. Aunque en algunos casos puntuales se observaron ligeras ganancias en métricas como el F1-score o el IoU, la mayoría de modelos sufrieron un pequeño descenso en el rendimiento. Esta situación puede atribuirse a varios factores:

- **Diferencias entre datasets**: INRIA contiene imágenes más homogéneas, urbanas y con delimitaciones de edificios más claras. En contraste, PNOA incluye una mayor variabilidad geográfica (entornos rurales, vegetación densa, sombras, etc.), lo que hace más difícil la generalización de los modelos.
- **Ruido en las máscaras del PNOA**: A diferencia del dataset INRIA, que cuenta con anotaciones más limpias, las máscaras de PNOA fueron generadas o ajustadas manualmente y presentan ruido (bordes imprecisos, errores de segmentación), lo que complica la tarea de ajuste fino.
- **Tamaño limitado del dataset**: Aunque el fine-tuning se realizó con tasas de aprendizaje bajas y estrategias de congelación, el conjunto de datos disponible puede no haber sido suficientemente representativo como para inducir una mejora significativa sin riesgo de sobreajuste.

Desde el punto de vista arquitectónico, se observa que los modelos más pesados (como DeepLabV3+ con ResNeXt101 o UPerNet con ConvNeXt) tienden a mantener mejor su rendimiento tras el fine-tuning. Estos modelos poseen una mayor capacidad de representación y parecen beneficiarse más de la adaptación, incluso cuando no se congelan capas (como fue el caso de UPerNet). En cambio, modelos más ligeros como PAN-MobileNetV2, si bien rápidos y eficientes, mostraron limitaciones para adaptarse al nuevo dominio, especialmente cuando el número de parámetros entrenables era reducido por la congelación parcial.

En cuanto a la estrategia de **congelamiento del encoder**, los resultados sugieren que su utilidad depende del equilibrio entre preservar conocimiento útil previo y permitir suficiente flexibilidad para aprender nuevas características. En algunos casos (como UNet-ResNet34), el congelamiento de capas iniciales permitió una adaptación estable sin sobreajuste. En otros (como PAN-MobileNetV2), pudo haber restringido demasiado la capacidad de adaptación, perjudicando el rendimiento final. Por el contrario, en UPerNet-ConvNeXt, al no congelarse ninguna capa, el modelo fue capaz de ajustarse completamente al dominio del PNOA, lo cual puede explicar su rendimiento competitivo.

En resumen, el fine-tuning aportó beneficios limitados y dependientes tanto del modelo como del contexto. Para obtener mejoras más consistentes, sería necesario o bien ampliar el dataset PNOA con más imágenes y anotaciones limpias, o adoptar estrategias más avanzadas como el semi-supervised learning o la adaptación de dominio explícita.

3.6 Entrenamiento conjunto con imágenes de PNOA e INRIA

Con el objetivo de estudiar si una mayor diversidad geográfica y visual en los datos mejora la capacidad de generalización del modelo, se diseñó un experimento adicional en el que se combinan imágenes de los conjuntos PNOA e INRIA para el entrenamiento de un único modelo. Para esta prueba se seleccionó el modelo UPerNet con backbone ConvNeXt_base, dado que en experimentos anteriores mostró un rendimiento competitivo y estable en ambos contextos.

Preparación de los datos

El preprocesamiento se adaptó para acomodar las diferencias entre los conjuntos de datos:

- **Máscaras binarizadas**: las máscaras de segmentación se escalaron al rango [0,1], en lugar del habitual [0,255], lo que permite tratar el problema como una segmentación binaria y facilita el uso de funciones de pérdida como Binary Cross Entropy.
- **Normalización de imágenes**:se utilizaron estadísticas de normalización específicas derivadas de los datos del PNOA y el IGN, con los siguientes valores:
  - **Media**: [0.372,0.362,0.344]
  - **Desviación estándar**: [0.307,0.294,0.281]
- **Transformaciones**: Se aplicaron aumentos de datos como HorizontalFlip, VerticalFlip, GridDistortion, RandomBrightnessContrast y GaussNoise, todos ellos con la librería Albumentations. Estos aumentos se usaron solo en el conjunto de entrenamiento para mejorar la capacidad de generalización del modelo.
- **División del dataset**:El conjunto combinado INRIA–PNOA se dividió en tres particiones para entrenamiento, validación y prueba, respetando una proporción clásica de:
  - **70 % para entrenamiento**

- **15 % para validación**
- **15 % para test**

Esta división se realizó de forma aleatoria pero reproducible (semilla fija random.seed(42)), asegurando una distribución representativa de ambas fuentes de datos en cada subconjunto. El total de muestras disponibles era de 27828 pares imagen-máscara, distribuidos de la siguiente manera:

- **Entrenamiento**: 19479 muestras
- **Validación**: 4174 muestras
- **Test**: 4175 muestras

Cada muestra consta de una imagen RGB y su correspondiente máscara binaria con valores entre 0 y 1.

Arquitectura y flujo de entrenamiento

El modelo empleado fue UPerNet-ConvNeXt_base con pesos de imagenet en el backbone.

Se introdujeron los siguientes ajustes clave respecto a experimentos anteriores:

- En este experimento se ha empleado la función de pérdida `CrossEntropyLoss`, proporcionada por la librería `PyTorch`. Esta función es ampliamente utilizada en tareas de clasificación multiclase, incluida la segmentación semántica, y resulta adecuada para problemas con dos o más clases. En este caso, dado que el problema de segmentación se ha tratado como una clasificación de dos clases (fondo vs edificio), se ha aplicado directamente sin modificación.

  La función `CrossEntropyLoss` combina internamente la operación de `LogSoftmax` con la pérdida de log-verosimilitud negativa (`NLLLoss`). Su expresión matemática, para una muestra con C clases, es:

  $$\mathcal{L}(x, y) = -\log\left(\frac{\exp(x_y)}{\sum_{j=0}^{C-1}\exp(x_j)}\right) = -x_y + \log\left(\sum_{j=0}^{C-1}\exp(x_j)\right)$$

  donde:

  - $x \in R^C$ representa el vector de logits (salida cruda de la red antes del `softmax`).
  - $y \in \{0, 1, \ldots, C-1\}$ es la clase verdadera.
  - C es el número de clases (en nuestro caso, C=2).

- Durante el entrenamiento del modelo conjunto con datos del PNOA e INRIA, se añadieron nuevas métricas de evaluación con el objetivo de hacer un seguimiento más preciso del rendimiento del modelo. En concreto, se utilizaron las siguientes:

  - **Precisión por píxel (Pixel Accuracy)**: Esta métrica mide el porcentaje de píxeles correctamente clasificados sobre el total:

  $$\text{Pixel Accuracy} = \frac{\sum_i \text{TP}_i}{\sum_i (\text{TP}_i + \text{FP}_i + \text{FN}_i + \text{TN}_i)}$$

En el caso binario (edificio vs. no edificio), puede simplificarse como:

$$\text{Pixel Accuracy} = \frac{\text{Nº píxeles bien clasificados}}{\text{Total de píxeles}}$$

- **Intersection over Union media (mIoU)**: La métrica mIoU mide la superposición entre la predicción y el valor real para cada clase, promediando entre ellas:

$$\text{IoU}_i = \frac{TP_i}{TP_i + FP_i + FN_i} \quad \Rightarrow \quad \text{mIoU} = \frac{1}{C}\sum_{i=1}^{C} \text{IoU}_i$$

- **Accuracy por clase y media (acc, acc_cls)**:
  - **Acc (global)**:

$$\text{Acc} = \frac{\sum_i TP_i}{\sum_i (\text{Todos los píxeles})}$$

  - **Acc por clase media (acc_cls)**:

$$\text{Acc\_cls} = \frac{1}{C}\sum_{i=1}^{C} \frac{TP_i}{TP_i + FN_i}$$

- **Accuracy ponderado por frecuencia (fwavacc)**: Esta métrica da más peso a las clases que aparecen más frecuentemente en las imágenes:

$$\text{fwavacc} = \sum_{i=1}^{C} f_i \cdot \text{IoU}_i$$

  donde f_i es la frecuencia de la clase i en el conjunto de datos.
- **Pérdida de validación (val_loss)**: Se utiliza el valor medio de la función de pérdida sobre todo el conjunto de validación para monitorizar el sobreajuste.

- **Optimización**: Se utilizó el optimizador AdamW con una tasa de aprendizaje inicial de $1 \cdot 10^{-}3$ y un weight decay de $1 \cdot 10^{-}4$, valores comunes en tareas de segmentación con arquitecturas pesadas.

- **Scheduler**: Se empleó el planificador de tasa de aprendizaje ReduceLROnPlateau, que monitoriza la pérdida de validación y reduce automáticamente la tasa de aprendizaje si no se observa mejora durante 5 épocas consecutivas. El factor de reducción fue de 0.1, y se estableció un valor mínimo de aprendizaje de $1 \cdot 10^{-}5$.

- **Early Stopping**: Para evitar un sobreentrenamiento innecesario, se incorporó un criterio de parada anticipada basado en la métrica de mIoU: si no se observaba mejora en 15 épocas seguidas, el entrenamiento se detenía automáticamente, guardando el mejor modelo observado. Se definió un número máximo de 250 épocas, si el proceso no llegaba a detenerse antes.

- **Batch size**: Se utilizaron 10 imágenes por lote durante el entrenamiento, mientras que para la validación se evaluó una imagen por lote para asegurar que se adaptaba a la memoria GPU disponible y evitar pérdidas de precisión por redimensionamientos adicionales.

Resultados de entrenamiento

Aunque no se registraron todas las épocas debido a interrupciones durante el entrenamiento, se dispone de registros de las últimas fases, que muestran una estabilidad y convergencia claras. A continuación se incluyen algunas de las gráficas extraídas del proceso:

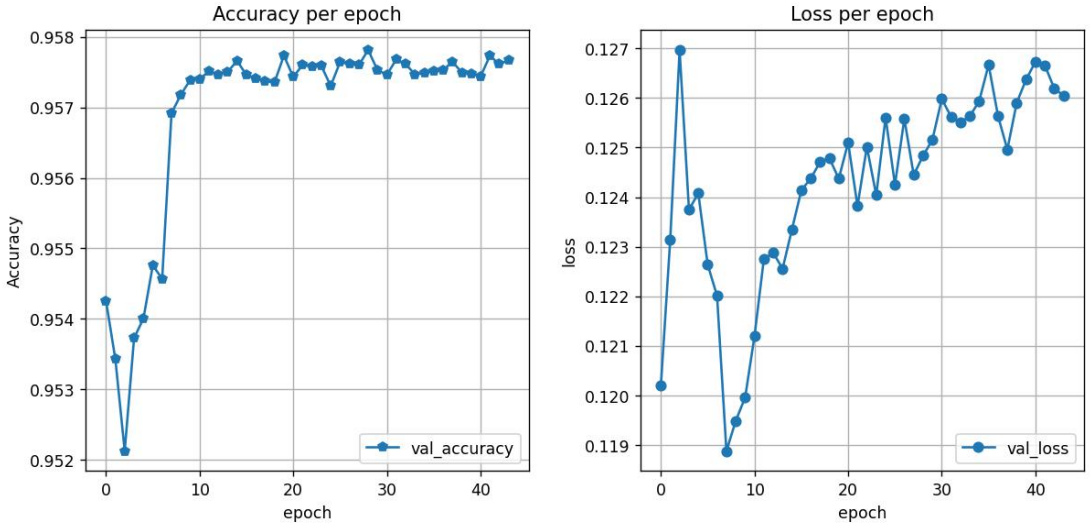


Figura 3.24: Evolución de la accuracy y pérdida de validación en las últimas épocas.

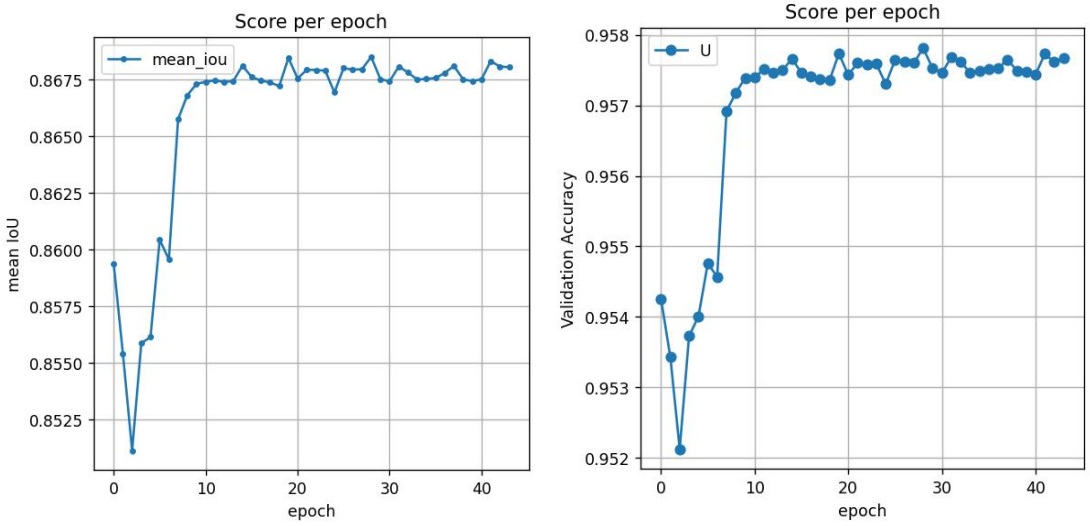


Figura 3.25: Curvas de IoU y Accuracy con `ReduceLROnPlateau` aplicado.

El entrenamiento conjunto de imágenes de INRIA y PNOA ha permitido al modelo mantener un rendimiento alto, superior al alcanzado al entrenar solo con PNOA. Esto sugiere que el modelo es capaz de aprender representaciones comunes a ambos dominios, siempre que se ajuste adecuadamente la normalización, las máscaras y la estructura del entrenamiento.

Los resultados mejoran significativamente respecto al fine-tuning con solo PNOA y muestran un comportamiento robusto, lo cual es especialmente valioso para tareas en las

que se requiere una generalización geográfica más amplia.

Capítulo 4

Resultados

4.1 Comparativa de modelos

Este apartado resume y compara el rendimiento de los modelos evaluados en las distintas fases del trabajo: entrenamiento sobre INRIA, fine-tuning con PNOA, y entrenamiento conjunto con ambos conjuntos de datos. El análisis se basa tanto en métricas cuantitativas (IoU, F1-score, etc.) como en observaciones cualitativas realizadas previamente.

Modelo	**INRIA**		**Fine-tuning PNOA**		**INRIA+PNOA**		
	IoU (%)	F1 (%)	IoU (%)	F1 (%)	mIoU (%)	F1 (%)	Acc (%)
UNet-ResNet34	74.7	-	71.7	83.2	-	-	-
DeepLabv3+-ResNeXt101	75.0	-	73.6	84.5	-	-	-
PAN-MobileNetV2	72.9	83.5	72.7	83.9	-	-	-
UPerNet-ConvNeXt	<u>78.2</u>	<u>87.1</u>	<u>75.0</u>	<u>85.4</u>	**86.7**	**92.6**	**95.7**

Cuadro 4.1: Comparativa de rendimiento por modelo y escenario de entrenamiento. Las métricas subrayadas representan el mejor valor dentro de cada escenario; los valores en negrita indican los mejores resultados globales.

Comparación por arquitectura

Los resultados reflejan claramente que las arquitecturas más profundas y modernas tienden a obtener mejores resultados. Por ejemplo:

- UPerNet-ConvNeXt Base se posiciona como el modelo más robusto, alcanzando altos valores de IoU y F1 tanto en el fine-tuning con PNOA como en el entrenamiento conjunto.
- DeepLabv3+ con ResNeXt101 también ofrece un rendimiento alto y estable, aunque ligeramente inferior al de UPerNet.
- UNet-ResNet34, siendo una arquitectura más simple, obtiene resultados razonables, pero menos competitivos en términos de generalización.
- PAN-MobileNetV2 presenta el rendimiento más bajo, aunque ofrece la ventaja de una baja complejidad computacional.

Generalización de los modelos

El entrenamiento sobre INRIA ofrecía buenos resultados sobre ese mismo dominio, pero en muchos casos, el fine-tuning sobre PNOA no mejoró las métricas. Esto sugiere que algunos modelos sufrieron overfitting o degradación del rendimiento al enfrentarse al nuevo dominio.

Influencia del backbone

Los backbones más modernos y profundos (como ConvNeXt o ResNeXt101) han demostrado una mayor capacidad de extracción de características relevantes para el problema de segmentación.

En cambio, los backbones más simples como ResNet34 o MobileNetV2 parecen limitados en su capacidad de adaptación a dominios variados o complejos como PNOA, en especial si hay ruido en las máscaras o diferencias espectrales en los datos.

Visualización gráfica de resultados

Para facilitar la comparación del rendimiento entre modelos y escenarios, se han generado representaciones gráficas de las métricas principales: **IoU** y **F1-score**. En estas gráficas se agrupan los resultados por modelo, mostrando la evolución de cada uno en los tres escenarios evaluados: entrenamiento con INRIA, fine-tuning con PNOA y entrenamiento conjunto con INRIA+PNOA.

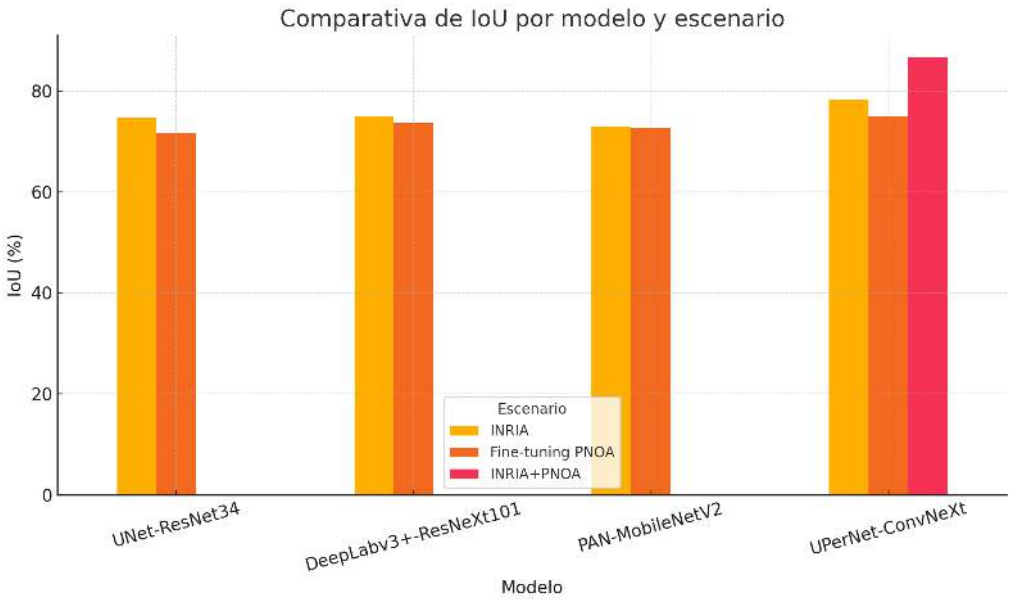


Figura 4.1: Comparación del IoU (%) por modelo en los diferentes escenarios.

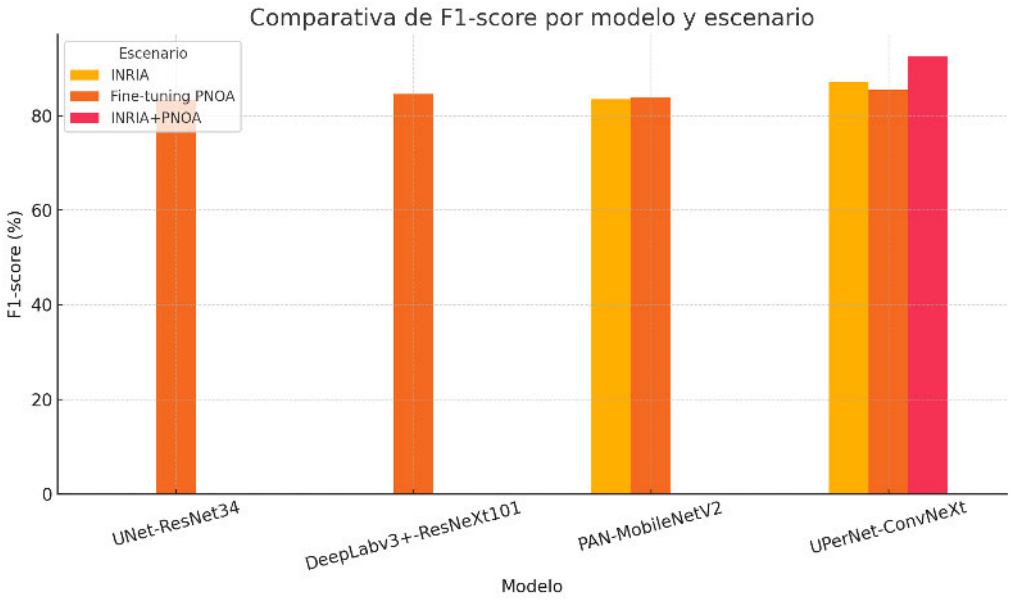


Figura 4.2: Comparación del F1-score (%) por modelo en los diferentes escenarios.

Estas visualizaciones permiten observar con claridad qué modelos muestran mejor capacidad de generalización, cómo afecta el fine-tuning en el rendimiento, y qué impacto tiene la combinación de datasets. Destaca el comportamiento consistente de UPerNet-ConvNeXt, que lidera en todos los escenarios, especialmente en el conjunto combinado INRIA+PNOA.

4.2 Evaluación sobre extensión geográfica real

Para comprobar la capacidad de generalización del modelo más potente entrenado, se evaluó el modelo UPerNet-ConvNeXt Base entrenado con el conjunto combinado INRIA+PNOA sobre una nueva área del territorio español nunca vista durante el entrenamiento: una región de aproximadamente 10×10 km^2 correspondiente al municipio de Móstoles, en la Comunidad de Madrid.

Proceso de inferencia y vectorización

El área fue procesada por el modelo previamente entrenado, generando una predicción segmentada con las zonas identificadas como edificios. Posteriormente, se llevó a cabo una vectorización de los resultados, transformando las máscaras binarias en polígonos mediante técnicas de contorneo. Para mejorar la calidad geométrica de los vectores resultantes, se aplicaron operaciones de generalización y rectificación, con el objetivo de eliminar artefactos y suavizar los contornos, adaptándolos mejor a formas urbanas reales.

El resultado final se guardó en un archivo vectorial (formato GeoPackage) para su visualización y análisis.

Visualización con QGIS

Para la visualización y evaluación visual, se utilizó el software QGIS (anteriormente conocido como "Quantum GIS"), una aplicación de código abierto ampliamente usada en

el análisis geoespacial y visualización de datos SIG (Sistemas de Información Geográfica) [28].

El archivo vectorial resultante se cargó como una capa en QGIS, y se superpuso a una capa base de ortoimagen proveniente de Google Satellite, lo que permitió una evaluación cualitativa del ajuste de los polígonos generados respecto a la imagen aérea real.

Evaluación cualitativa y ejemplos

A continuación, se muestran dos ejemplos representativos de los resultados obtenidos:

- Un caso positivo, donde el modelo detectó correctamente la mayoría de los edificios de una zona residencial, incluyendo contornos bien definidos y pocos errores de omisión o comisión.
- Un caso negativo, correspondiente a una zona con estructuras irregulares, donde el modelo cometió errores al confundir vegetación o caminos con construcciones.

Figura 4.3: Ejemplos de evaluación sobre territorio real. Izquierda: ejemplo positivo en zona urbana. Derecha: ejemplo negativo en zona compleja.

Discusión sobre errores y limitaciones

Se observaron patrones interesantes durante la evaluación:

- El modelo funcionó de forma sólida en zonas urbanas densas, donde los edificios tienen formas regulares y bien delimitadas.
- Sin embargo, se observó un mayor número de errores en áreas rurales o periféricas, donde los objetos pueden tener formas ambiguas o donde predominan estructuras no residenciales.
- También se detectaron inconsistencias causadas por posibles errores en las etiquetas de entrenamiento, especialmente provenientes de OpenStreetMap (OSM), lo cual pudo influir negativamente en la calidad del aprendizaje. Algunas etiquetas de edificios estaban ausentes o mal posicionadas, lo que genera confusión durante la inferencia.

Estos resultados reflejan la importancia de una anotación precisa y variada, así como de un diseño cuidadoso del conjunto de entrenamiento si se desea lograr una generalización efectiva en escenarios reales y heterogéneos.

4.3 Discusión de resultados

Reflexión global

El presente trabajo ha permitido analizar el desempeño de diferentes arquitecturas de segmentación semántica aplicadas a la detección de edificaciones sobre imágenes aéreas. A través de un riguroso proceso de entrenamiento, validación cruzada y evaluación geográfica real, se ha podido extraer una serie de conclusiones relevantes respecto a la capacidad de generalización de los modelos, el impacto del conjunto de datos, y los retos asociados a la implementación práctica de estas técnicas en contextos reales.

Retos principales

Uno de los principales desafíos identificados ha sido la disparidad entre los conjuntos de datos INRIA y PNOA. Mientras que INRIA presenta imágenes homogéneas con una anotación coherente y simplificada, el conjunto PNOA incluye una mayor variabilidad visual (en términos de iluminación, resolución y morfología urbana) y presenta más ruido en las máscaras, lo que complica el proceso de aprendizaje.

Durante el trabajo se evidenció la necesidad de realizar una validación manual de las máscaras derivadas del PNOA, ya que estas provenían en parte de fuentes como OpenStreetMap (OSM), las cuales no siempre son precisas ni completas. Se detectaron errores de omisión, duplicación o desplazamiento en los vectores de los edificios, lo que afecta directamente al proceso de entrenamiento. Este aspecto se volvió especialmente relevante al comparar el comportamiento de los modelos entrenados exclusivamente con INRIA frente a aquellos ajustados con PNOA o INRIA+PNOA.

Modelos con mayor potencial

De todos los modelos analizados, UPerNet con backbone ConvNeXt ha demostrado consistentemente ser el más robusto y preciso, tanto en métricas cuantitativas como en la evaluación visual sobre zonas no vistas. Su capacidad para mantener un rendimiento alto incluso en condiciones variadas lo posiciona como un modelo con gran potencial para aplicaciones reales, especialmente cuando se entrena sobre un conjunto mixto como INRIA+PNOA.

También se observaron buenos resultados con DeepLabv3+ con ResNeXt. No obstante, su desempeño general fue algo más sensible al cambio de dominio que el de UPerNet.

Comparación entre teoría y práctica

Si bien los resultados en métricas como IoU o F1 fueron útiles como indicadores generales, la evaluación práctica en mapas reales (por ejemplo, en la zona de Móstoles) reveló que incluso modelos con alta puntuación pueden cometer errores relevantes desde una perspectiva cartográfica o urbanística. Esto demuestra que, además de buenas métricas

en conjuntos de validación, es fundamental realizar evaluaciones visuales o manuales sobre escenarios reales para determinar la verdadera utilidad de los modelos.

Implicaciones para el despliegue

Los resultados alcanzados abren la puerta a aplicaciones reales en contextos como:

- Urbanismo y ordenación del territorio, donde se podría automatizar parcialmente la actualización de cartografía de edificaciones.
- Gestión de emergencias y catástrofes, al permitir una detección rápida de estructuras afectadas en imágenes aéreas recientes.
- Monitoreo de crecimiento urbano, mediante análisis periódicos de zonas en expansión.

No obstante, para un despliegue robusto en sistemas productivos, se requieren más esfuerzos en la curación de los datos, la validación sistemática de los resultados y la integración con plataformas geográficas como QGIS o servicios web de mapas.

Capítulo 5

Conclusiones

Este trabajo ha tenido como objetivo principal el desarrollo y comparación de diversos modelos de segmentación semántica orientados a la detección automática de edificaciones en imágenes aéreas RGB, con el fin de contribuir a la mejora de procesos de análisis espacial en aplicaciones como la cartografía, la ordenación territorial o la gestión urbanística.

A lo largo del proyecto, se han abordado diferentes objetivos específicos: desde la revisión del estado del arte en arquitecturas modernas de segmentación, hasta el diseño y curación de un dataset personalizado a partir de imágenes del PNOA, pasando por el entrenamiento de modelos sobre el conjunto INRIA, la aplicación de técnicas de transfer learning, la evaluación cuantitativa y cualitativa de resultados, y la reflexión sobre el potencial impacto de estas tecnologías en contextos reales.

Principales hallazgos

- La arquitectura UPerNet con backbone ConvNeXt ha demostrado ser la más robusta y precisa en todos los escenarios evaluados, tanto en métricas como en la evaluación sobre datos reales.
- El uso del dataset INRIA como base de preentrenamiento no proporciona una gran ventaja inicial, comprobado cuando se ha aplicado un fine-tuning con imágenes del dominio español (PNOA).
- El proceso de creación de máscaras precisas para el conjunto PNOA resultó ser crítico para obtener buenos resultados, ya que los errores en las máscaras derivadas de OSM impactan negativamente en el rendimiento del modelo.
- El transfer learning ha permitido adaptar modelos preentrenados a contextos diferentes con buenos resultados, especialmente cuando se cuenta con una cantidad limitada de datos anotados localmente.

Comparativa de modelos

Aunque todos los modelos han sido capaces de aprender representaciones útiles, se ha observado que las arquitecturas más profundas o modernas (como UPerNet-ConvNeXt) ofrecen mejor generalización y precisión, mientras que modelos más ligeros (como PAN-MobileNetV2) ofrecen una opción más eficiente, pero con menor rendimiento en métricas como IoU o F1.

Evaluación de los datos

Una conclusión clave ha sido la gran diferencia entre la calidad de los datos de INRIA y los generados a partir de OSM para el PNOA. Mientras que INRIA ofrece máscaras coherentes y homogéneas, los datos de OSM presentan errores frecuentes como omisiones, duplicidades o desalineaciones, que requieren validación y corrección manual para evitar que los modelos aprendan patrones incorrectos.

Valoración final del proyecto

El proyecto ha permitido desarrollar un sistema funcional capaz de detectar edificaciones sobre imágenes aéreas con un alto nivel de precisión, aprovechando tanto técnicas modernas de aprendizaje profundo como herramientas de visualización geoespacial (QGIS). Además, se ha logrado demostrar la viabilidad de la segmentación semántica como herramienta de apoyo para tareas urbanísticas, con modelos que pueden generalizar a nuevas regiones geográficas con un margen razonable de error.

A nivel crítico, el trabajo pone de manifiesto que la calidad y homogeneidad del dato sigue siendo el mayor reto en aplicaciones reales. Aunque las técnicas de deep learning avanzan rápidamente, su eficacia depende en gran medida de la fiabilidad del etiquetado. Por ello, una línea futura imprescindible sería la automatización de procesos de validación y mejora de máscaras, así como el uso de enfoques semi-supervisados o active learning.

En conclusión, el trabajo ha cumplido con los objetivos planteados, ha generado un modelo útil y ha dejado planteadas varias líneas claras para investigaciones futuras en el campo de la inteligencia artificial aplicada a la geoinformación.

5.1 Futuras líneas de trabajo

A partir de los resultados obtenidos y de las limitaciones encontradas durante el desarrollo del proyecto, se identifican varias líneas prometedoras de trabajo futuro que permitirían seguir mejorando la precisión, generalización y aplicabilidad de los modelos desarrollados.

Exploración de nuevas arquitecturas

Una evolución natural de este trabajo es la evaluación de nuevas arquitecturas de segmentación semántica, particularmente aquellas basadas en transformers y modelos híbridos. Arquitecturas como Swin Transformer, SegFormer o Mask2Former han demostrado un rendimiento sobresaliente en benchmarks recientes y podrían mejorar significativamente la capacidad del modelo para captar relaciones espaciales complejas y detalles finos, incluso en contextos con poca señal como zonas rurales o con objetos parcialmente visibles.

También resulta interesante explorar modelos más eficientes y ligeros, como Fast-SCNN, BiSeNet o variantes móviles de SegFormer, que permitirían desplegar los modelos en dispositivos con recursos limitados, como móviles, drones o sistemas embebidos.

Mejora y ampliación de los datos

Otra línea crítica consiste en ampliar el conjunto de entrenamiento con más imágenes reales y máscaras de calidad, especialmente del territorio español. La calidad del etiquetado en los datos derivados de OpenStreetMap (OSM) ha demostrado ser un cuello de botella, por lo que se propone desarrollar estrategias para:

- La arquitectura UPerNet con backbone ConvNeXt ha demostrado ser la más robusta y precisa en todos los escenarios evaluados, tanto en métricas como en la evaluación sobre datos reales.
- Automatizar la validación y corrección de máscaras mediante heurísticas geométricas, aprendizaje semi-supervisado o asistencia humana supervisada.
- Aprovechar técnicas de data augmentation avanzadas o síntesis de datos para simular variabilidad geográfica y estructural.
- Incluir nuevas zonas geográficas con diversidad morfológica (zonas rurales, zonas costeras, islas, etc.) que pongan a prueba la capacidad del modelo de adaptarse a distintas realidades espaciales.

Aplicaciones futuras

Los modelos desarrollados tienen un enorme potencial en diversas aplicaciones prácticas, algunas de las cuales podrían constituir líneas futuras de desarrollo:

- **Análisis temporal y detección de cambios:**detectar nuevas construcciones o demoliciones mediante el análisis de imágenes de diferentes años.
- **Apoyo a la planificación urbana y gestión del territorio**, generando mapas base automáticos en zonas sin cartografía precisa.
- **Evaluación del impacto climático o de catástrofes naturales**, analizando la evolución de la urbanización o la afectación de zonas edificadas tras un evento extremo.
- **Desarrollo de sistemas de mapeo colaborativo** que aprovechen predicciones automáticas para acelerar la edición de bases de datos geográficas abiertas como OSM.

Despliegue en sistemas reales

Finalmente, una dirección estratégica a largo plazo es la implementación de estos modelos en sistemas productivos reales. Esto podría incluir:

- Aplicaciones web o móviles que permitan la detección de edificaciones a partir de imágenes aéreas cargadas por el usuario.
- Integración en flujos de trabajo de organismos públicos de urbanismo o catastro.
- Soporte para tareas de emergencia (por ejemplo, mapeo rápido tras desastres naturales).

Este tipo de implementación requerirá trabajar aspectos como la interfaz, la escalabilidad, la eficiencia y la explicabilidad del modelo, lo que abre nuevas áreas de investigación técnico-práctica.

5.2 Impacto social y medioambiental

El desarrollo de sistemas automáticos para la detección de edificaciones a partir de imágenes aéreas tiene un alto potencial de impacto positivo, tanto a nivel social como medioambiental, si se integran de forma responsable y ética en aplicaciones reales.

Aplicaciones en planificación urbana y ordenación territorial

Uno de los principales ámbitos de aplicación es la ordenación urbana y la planificación sostenible del territorio. La automatización en la identificación de edificios permite:

- Detectar expansiones urbanas no registradas o construcciones ilegales.
- Actualizar mapas catastrales con mayor rapidez y frecuencia.
- Identificar zonas vulnerables al hacinamiento o al crecimiento descontrolado.

Este tipo de análisis ofrece a los gobiernos locales y autonómicos una herramienta valiosa para la toma de decisiones basada en datos, lo que puede mejorar la equidad territorial, la eficiencia en la distribución de recursos y la sostenibilidad del crecimiento urbano.

Apoyo en situaciones de emergencia

En contextos de emergencias naturales, como terremotos, incendios o inundaciones, disponer de modelos entrenados para detectar rápidamente zonas edificadas puede ser fundamental para:

- Mapear áreas afectadas en cuestión de minutos tras un desastre.
- Identificar edificaciones destruidas o aisladas.
- Coordinar planes de evacuación o rescate, especialmente en zonas rurales o con baja cobertura cartográfica.

La capacidad de generalización demostrada por algunos modelos en este trabajo, especialmente UPerNet-ConvNeXt, sugiere una posible utilidad práctica en estos escenarios críticos.

Sostenibilidad y resiliencia climática

Estos modelos también pueden ser utilizados como herramienta para monitorear la expansión urbana sobre zonas verdes o agrícolas, facilitando políticas de protección ambiental y control del suelo. De igual modo, ayudan a modelar el impacto urbano en fenómenos climáticos como:

- Islas de calor urbanas.

- Riesgos de inundación por urbanización excesiva.
- Vulnerabilidad estructural frente a fenómenos extremos.

Esto contribuye al diseño de ciudades más resilientes frente al cambio climático.

Ética y privacidad en el uso de datos geoespaciales

El uso de imágenes aéreas y modelos de segmentación plantea también desafíos éticos que deben ser abordados cuidadosamente. Aunque las imágenes utilizadas (como las del PNOA o Google Satellite) son generalmente de acceso público, el uso automatizado de estas para extraer información sensible (como la localización de edificaciones privadas) puede generar desconfianza sobre usos indebidos, privacidad o vigilancia.

Es fundamental asegurar que:

- Los datos sean empleados exclusivamente con fines públicos, científicos o de mejora territorial.
- Se cumplan las normativas de protección de datos y confidencialidad.
- Se eviten usos comerciales o de vigilancia no autorizada de estos sistemas.

Asimismo, el diseño de los modelos debe contemplar principios de transparencia, auditabilidad y explicabilidad, garantizando que las predicciones puedan ser entendidas y revisadas por humanos.

El código implementado y los resultados obtenidos están disponibles en el siguiente repositorio de Github [29].

Bibliografía

[1] Alexander Buslaev, Vladimir I Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A Kalinin. Albumentations: fast and flexible image augmentations. *Information*, 11(2):125, 2020.

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[3] Miguel Angel Chicchón Apaza. Fusión de datos para segmentación semántica en aplicaciones urbanas de teledetección aérea usando algoritmos de aprendizaje profundo. 2018.

[4] Calimanut-Ionut Cira, Miguel-Ángel Manso-Callejo, Ramón Alcarria, Borja Bordel Sánchez, and Javier González Matesanz. State-level mapping of the road transport network from aerial orthophotography: an end-to-end road extraction solution based on deep learning models trained for recognition, semantic segmentation and post-processing with conditional generative learning. *Remote Sensing*, 15(8):2099, 2023.

[5] Alex Clark et al. Pillow (pil fork) documentation. *readthedocs*, 2015.

[6] OpenStreetMap contributors. Openstreetmap: datos geoespaciales abiertos, 2025.

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[9] Jaime Duque Domingo, Roberto Medina Aparicio, and Luis Miguel Gonzalez Rodrigo. Improvement of one-shot-learning by integrating a convolutional neural network and an image descriptor into a siamese neural network. *Applied Sciences*, 11(17):7839, 2021.

[10] Omar Elharrouss, Younes Akbari, Noor Almadeed, and Somaya Al-Maadeed. Backbones-review: Feature extractor networks for deep learning and deep reinforcement learning approaches in computer vision. *Computer Science Review*, 53:100645, 2024.

[11] Florian L Faltermeier, Sebastian Krapf, Bruno Willenborg, and Thomas H Kolbe. Improving semantic segmentation of roof segments using large-scale datasets derived from 3d city models and high-resolution aerial imagery. *Remote Sensing*, 15(7):1931, 2023.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] IBM. What is Semantic Segmentation? `https://www.ibm.com/think/topics/semantic-segmentation`, 2021.

[14] Vladimir Iglovikov and Alexey Shvets. Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation. *arXiv preprint arXiv:1801.05746*, 2018.

[15] Fabian Isensee, Jens Petersen, Andre Klein, David Zimmerer, Paul F Jaeger, Simon Kohl, Jakob Wasserthal, Gregor Koehler, Tobias Norajitra, Sebastian Wirkert, et al. nnu-net: Self-adapting framework for u-net-based medical image segmentation. *arXiv preprint arXiv:1809.10486*, 2018.

[16] Ankit Kariryaa. Maskit: Masking for efficient utilization of incomplete public datasets for training deep learning models. *arXiv preprint arXiv:2006.12004*, 2020.

[17] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9404–9413, 2019.

[18] Hyunkwang Lee, Fabian M Troschel, Shahein Tajmir, Georg Fuchs, Julia Mario, Florian J Fintelmann, and Synho Do. Pixel-level deep segmentation: artificial intelligence quantifies muscle on computed tomography for body morphometric analysis. *Journal of digital imaging*, 30:487–498, 2017.

[19] Hanchao Li, Pengfei Xiong, Jie An, and Lingxue Wang. Pyramid attention network for semantic segmentation. *arXiv preprint arXiv:1805.10180*, 2018.

[20] Yingjian Li, Yonggang Li, Xiangbin Zhu, Haojie Fang, and Lihua Ye. A method for extracting buildings from remote sensing images based on 3dja-unet3+. *Scientific Reports*, 14(1):19067, 2024.

[21] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[22] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[23] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[24] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In *2017 IEEE International geoscience and remote sensing symposium (IGARSS)*, pages 3226–3229. IEEE, 2017.

[25] Dushyant Mehta, Andrii Skliar, Haitam Ben Yahia, Shubhankar Borse, Fatih Porikli, Amirhossein Habibian, and Tijmen Blankevoort. Simple and efficient architectures for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2628–2636, 2022.

[26] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. Ieee, 2016.

[27] Sanjida Afrin Mou, Tasfia Noor Chowdhury, Adib Ibn Mannan, Sadia Nourin Mim, Lubana Tarannum, Tasrin Noman, and Jamal Uddin Ahamed. Ai driven water segmentation with deep learning models for enhanced flood monitoring. *arXiv preprint arXiv:2501.08266*, 2025.

[28] Nicolas Moyroud and Frédéric Portet. Introduction to qgis. *QGIS and generic tools*, 1:1–17, 2018.

[29] Jorge Pastor. semantic-building-segmentation. `https://github.com/pastooor/semantic-building-segmentation`, 2025.

[30] Proyecto Jupyter. Jupyter notebook: Entorno interactivo para computación científica, 2025.

[31] Md Atiqur Rahman and Yang Wang. Optimizing intersection-over-union in deep neural networks for image segmentation. In *International symposium on visual computing*, pages 234–244. Springer, 2016.

[32] Leo Thomas Ramos and Angel D Sappa. Leveraging u-net and selective feature extraction for land cover classification using remote sensing imagery. *Scientific Reports*, 15(1):784, 2025.

[33] Damien Rolon-Mérette, Matt Ross, Thaddé Rolon-Mérette, and Kinsey Church. Introduction to anaconda and python: Installation and setup. *Quant. Methods Psychol*, 16(5):S3–S11, 2016.

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[35] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[36] Yihao Sun, Mingrui Wang, Xiaoyi Huang, Chengshu Xin, and Yinan Sun. Fast semantic segmentation of ultra-high-resolution remote sensing images via score map and fast transformer-based fusion. *Remote Sensing*, 16(17):3248, 2024.

[37] Maofeng Tang, Konstantinos Georgiou, Hairong Qi, Cody Champion, and Marc Bosch. Semantic segmentation in aerial imagery using multi-level contrastive learning with local consistency. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3798–3807, 2023.

[38] Claudio Urrea and Maximiliano Vélez. Advances in deep learning for semantic segmentation of low-contrast images: A systematic review of methods, challenges, and future directions. *Sensors*, 25(7):2043, 2025.

[39] Guido Van Rossum et al. Python programming language. In *USENIX annual technical conference*, volume 41, pages 1–36. Santa Clara, CA, 2007.

[40] Esther Vera Moreno. Detección satelital de edificios y sus aplicaciones mediante redes neuronales. 2022.

[41] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 418–434, 2018.

[42] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[43] William Falcon y el equipo de PyTorch Lightning. Pytorch lightning: Librería para entrenamiento eficiente de modelos en pytorch, 2025.

[44] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9:611–629, 2018.

[45] Yaning Yi, Zhijie Zhang, Wanchang Zhang, Chuanrong Zhang, Weidong Li, and Tian Zhao. Semantic segmentation of urban buildings from vhr remote sensing imagery using a deep convolutional neural network. *Remote sensing*, 11(15):1774, 2019.

[46] Hua Zhang, Zhengang Jiang, Guoxun Zheng, and Xuekun Yao. Semantic segmentation of high-resolution remote sensing images with improved u-net based on transfer learning. *International Journal of Computational Intelligence Systems*, 16(1):181, 2023.

[47] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[48] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.