ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Máster Universitario en Big Data y Analítica Avanzada

Identificación y priorización inteligente de hogares para energía solar

Autor
Leticia Cólogan Valero

Dirigido por
Ana Laguna Pradas

Madrid
Junio 2025

Nombre del Autor, declara bajo su responsabilidad, que el Proyecto con título **TÍTULO DEL PROYECTO** presentado en la ETS de Ingeniería (ICAI) de la Universidad Pontificia Comillas en el curso académico 2024/25 es de su autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Leticia Cólogan Fecha: 24 / Junio / 2025

Autoriza la entrega:

El Director del Proyecto

Ana Laguna Pradas

Fdo.: ..... Fecha: 24 / Junio / 2025

V. B. del Coordinador de Proyectos

Carlos Morrás Ruiz-Falcó

Fdo.: ..... Fecha: ..... / ..... / .....

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D.Leticia Cólogan Valero **DECLARA** ser el titular de los derechos de propiedad intelectual de la obra: Identificación y priorización inteligente de hogares para energía solar, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, los derechos de digitalización, de archivo, de reproducción, de distribución y de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

(a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.

(b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.

(c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.

(d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.

(e) Asignar por defecto a estos trabajos una licencia Creative Commons.

(f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente*).

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

(a) Que la Universidad identifique claramente su nombre como autor de la misma

(b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.

(c) Solicitar la retirada de la obra del repositorio por causa justificada.

(d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

(a) El autor se compromete a:

(b) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.

(c) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.

(d) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

(e) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 24 de Junio de 2025.

ACEPTA

Fdo.: Leticia Cólogan

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Máster Universitario en Big Data y Analítica Avanzada

Identificación y priorización inteligente de hogares para energía solar

Autor
Leticia Cólogan Valero

Dirigido por
Ana Laguna Pradas

Madrid
Junio 2025

RESUMEN

Este trabajo desarrolla un sistema automático para la detección de instalaciones fotovoltaicas en imágenes aéreas, utilizando técnicas de segmentación semántica basadas en redes neuronales convolucionales. Se han explorado tres variantes de la arquitectura U-Net, incluyendo una versión ligera y otra con *transfer learning* (ResNet34). Los resultados muestran que la U-Net estándar logra el mejor rendimiento en el conjunto de validación original, mientras que el modelo con ResNet34 ofrece mejores resultados en un nuevo dominio geográfico, aunque con segmentaciones más difusas.

Palabras clave: Segmentación semántica, paneles solares, U-Net, imágenes aéreas, aprendizaje profundo.

1. Introducción

La transición energética hacia fuentes renovables ha impulsado el desarrollo de soluciones basadas en Deep Learning, Big Data e imágenes aéreas para identificar automáticamente la presencia de instalaciones solares en entornos urbanos.

La energía fotovoltaica, impulsada por avances tecnológicos y políticas de apoyo institucional, requiere métodos eficientes para monitorizar su despliegue y detectar nuevas oportunidades.

En este contexto, el uso de algoritmos de visión por computadora aplicados a imágenes satelitales se presenta como una herramienta clave para la detección masiva de paneles solares en tejados residenciales. Este enfoque permite optimizar procesos como la planificación energética, el análisis de impacto medioambiental y la priorización de áreas con alto potencial solar.

Este proyecto desarrolla una solución basada en modelos de segmentación semántica, entrenados con arquitecturas tipo U-Net, para identificar de forma precisa y automática tejados con instalaciones solares.

2. Definición del proyecto

El presente trabajo se basa en el entrenamiento de modelos de segmentación semántica sobre un conjunto de datos público compuesto por imágenes aéreas de tejados en distintas regiones de Francia, que incluyen tanto las imágenes originales como sus respectivas máscaras de segmentación.

Con el fin de evaluar la capacidad de generalización de los modelos, se ha empleado un conjunto independiente de imágenes reales del municipio de Molina de Segura (Murcia), las cuales han sido anotadas manualmente para su uso en la fase de validación.

Se han comparado tres variantes de la arquitectura U-Net:

- **U-Net estándar**: Entrenada desde cero con todos los parámetros inicializados aleatoriamente.
- **Mini U-Net**: Una versión más liviana y eficiente, diseñada para entornos con recursos computacionales limitados.
- **U-Net + ResNet34**: Arquitectura híbrida que incorpora un encoder preentrenado en ImageNet mediante técnicas de *transfer learning*.

3. Descripción del sistema

El proceso comienza con una fase de extracción de características a partir de imágenes aéreas RGB utilizando una red convolucional preentrenada (ResNet50). Esta red, utilizada como extractor sin su capa de clasificación final, genera vectores de características de alta dimensión que capturan información visual relevante (texturas, formas, estructuras).

Posteriormente, se aplica un algoritmo de clustering no supervisado (K-Means) sobre estos vectores para agrupar las imágenes según su similitud visual. Este paso permite filtrar automáticamente aquellas imágenes que no contienen construcciones residenciales (ej. campos, carreteras o zonas industriales), conservando solo las que muestran viviendas unifamiliares, que son el foco de estudio para la detección de paneles solares.

Una vez depurado el conjunto de datos, se procede al entrenamiento de los modelos de segmentación. En concreto, se han entrenado tres arquitecturas distintas: U-Net estándar, Mini U-Net y U-Net con codificador ResNet34 preentrenado.

Cada uno ha sido entrenado desde cero o mediante técnicas de transfer learning, según su configuración.

El entrenamiento se ha llevado a cabo en Google Colab, empleando técnicas como `early stopping` y evaluación mediante métricas estándar como Dice y IoU. Además, se ha utilizado Grad-CAM para interpretar visualmente la atención del modelo durante la inferencia, ayudando a entender en qué zonas de la imagen se concentra la detección.

4. Resultados

Los resultados obtenidos permiten comparar el rendimiento de los tres modelos tanto en el conjunto de validación original como en un entorno real distinto:

En el conjunto de validación (dataset francés), la U-Net estándar obtiene el mejor rendimiento cuantitativo, con un coeficiente Dice de 0.9058, seguida de la Mini U-Net (0.8863) y la U-Net con encoder ResNet34 (0.8653).

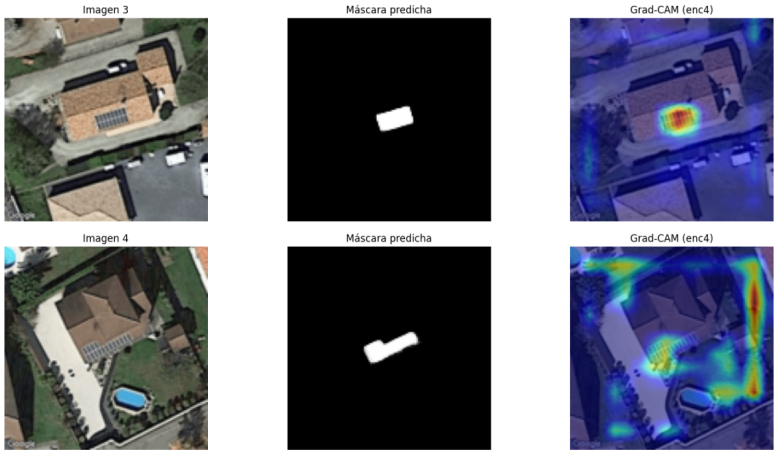

Figura 1: Predicción y mapa de activación del modelo **UNet** sobre dos ejemplos.

En las imágenes reales del municipio de Molina de Segura, el modelo con ResNet34 destaca con un Dice de 0.6620, mientras que la U-Net estándar y la Mini U-Net apenas alcanzan 0.0278, lo que sugiere una pobre generalización. No obstante, las segmentaciones del modelo con ResNet34 presentan menor definición y bordes más difusos.

Figura 2: Predicción de U-Net +ResNet34 sobre una imagen de Molina de Segura (imagen, máscara predicha).

Estos resultados reflejan la dificultad de generalizar a nuevos dominios geográficos, incluso cuando se emplean modelos avanzados. Aunque el transfer learning mejora las métricas, no siempre se traduce en una segmentación útil.

Cuadro 1: Comparativa de rendimiento de modelos U-Net

Modelo	**Dice (Validación)**	**Dice (Imágenes reales)**
U-Net estándar	0.9058	0.0278
Mini U-Net	0.8863	0.0278
U-Net + ResNet34	0.8653	**0.6620**

5. Conclusiones

Los resultados obtenidos muestran que la arquitectura U-Net estándar proporciona la mayor precisión en el conjunto de validación original, mientras que el modelo basado en ResNet34, gracias al uso de transfer learning, presenta una mejor capacidad de generalización en entornos no vistos, aunque con segmentaciones menos definidas. Por su parte, la Mini U-Net representa una alternativa eficiente para escenarios con recursos computacionales limitados.

Este trabajo valida el potencial de las técnicas de segmentación semántica para el análisis automatizado del despliegue fotovoltaico en entornos urbanos. Además, pone de relieve la importancia de complementar las métricas cuantitativas con evaluaciones cualitativas (por ejemplo, mediante Grad-CAM), especialmente bajo cambios de dominio.

Como líneas futuras de mejora, se proponen:

- Ampliar el dataset con imágenes reales y segmentaciones locales más representativas.
- Refinar el encoder con técnicas de fine-tuning adaptadas a tejados del sur de Europa.
- Integrar fuentes complementarias como datos catastrales, mapas solares o series temporales.
- Explorar arquitecturas más recientes (p.ej. Transformers para visión o Swin-UNet).

En conjunto, este proyecto constituye una base sólida para el desarrollo de herramientas inteligentes de planificación energética territorial y análisis de autoconsumo solar a gran escala.

6. Referencias

- https://github.com/leticiacologan/TFM

ABSTRACT

This project presents an automated system for detecting photovoltaic installations in aerial imagery, using semantic segmentation techniques based on convolutional neural networks. Three variants of the U-Net architecture have been explored, including a lightweight version and another based on *transfer learning* (ResNet34). Results show that the standard U-Net achieves the best performance on the original validation set, while the ResNet34 model yields better outcomes in a new geographic domain, albeit with blurrier segmentations.

Keywords: Semantic segmentation, solar panels, U-Net, aerial imagery, deep learning.

1. Introduction

The energy transition towards renewable sources has driven the development of solutions based on Deep Learning, Big Data, and aerial imagery to automatically detect solar installations in urban environments.

Photovoltaic energy, boosted by technological advances and institutional policies, requires efficient methods to monitor its deployment and identify new opportunities.

In this context, the use of computer vision algorithms applied to satellite images emerges as a key tool for large-scale detection of solar panels on residential rooftops. This approach helps optimize processes such as energy planning, environmental impact analysis, and the prioritization of high-potential solar areas.

This project develops a solution based on semantic segmentation models, trained with U-Net architectures, to accurately and automatically identify rooftops with solar installations.

2. Project Definition

This Project is based on training semantic segmentation models using a public dataset composed of aerial images of rooftops from various regions in France, which include both original images and their associated segmentation masks.

To evaluate the generalization capacity of the models, an independent dataset of real images from the municipality of Molina de Segura (Murcia, Spain) was used. These images were manually annotated and employed for the validation phase.

Three U-Net architecture variants were compared:

- **Standard U-Net**: Trained from scratch with all parameters randomly initialized.
- **Mini U-Net**: A lighter, more efficient version designed for environments with limited computational resources.
- **U-Net + ResNet34**: A hybrid architecture incorporating a pre-trained encoder on ImageNet via *transfer learning*.

3. System Description

The process begins with a feature extraction phase from RGB aerial images using a pre-trained convolutional neural network (ResNet50). This network, used without its final classification layer, generates high-dimensional feature vectors that capture relevant visual information (textures, shapes, structures).

Subsequently, an unsupervised clustering algorithm (K-Means) is applied to group images based on their visual similarity. This step allows for automatic filtering of images that do not contain residential buildings (e.g., fields, roads, industrial areas), retaining only those with detached houses, which are the focus of the solar panel detection task.

After filtering, the training of the segmentation models begins. Three different architectures were trained: standard U-Net, Mini U-Net, and U-Net with a pre-trained ResNet34 encoder. Each model was trained either from scratch or using transfer learning, depending on its configuration.

Training was conducted on Google Colab, employing techniques such as `early stopping` and performance evaluation using standard metrics like Dice and IoU.

Additionally, Grad-CAM was applied to visually interpret the model's attention during inference, helping to understand which image regions were most influential.

4. Results

The results allow for a comparison of the three models' performance, both on the original validation set and in a new real-world setting:

On the validation set (French dataset), the standard U-Net achieved the highest quantitative performance with a Dice coefficient of 0.9058, followed by Mini U-Net (0.8863) and U-Net + ResNet34 (0.8653).

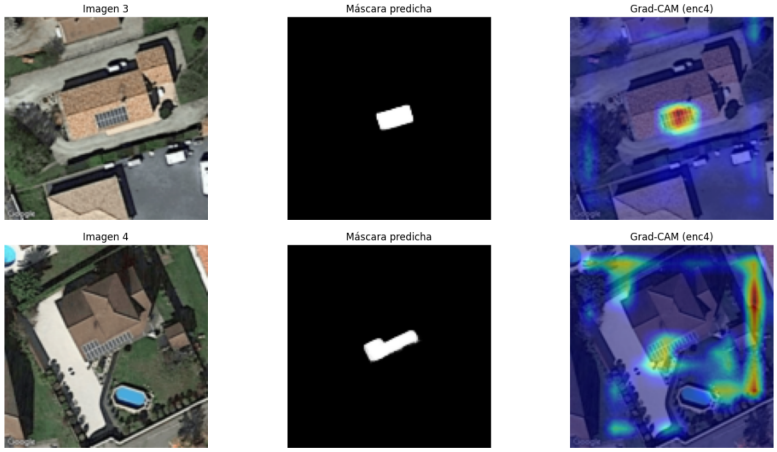

Figura 3: Prediction and Grad-CAM activation maps from the **U-Net** model on two examples.

On real-world images from Molina de Segura, the ResNet34 model stood out with a Dice score of 0.6620, while both the standard and Mini U-Net models achieved only 0.0278, indicating poor generalization. However, the segmentations generated by the ResNet34 model were less sharp and more diffuse.

Figura 4: Prediction from the U-Net + ResNet34 model on a real image (input, predicted mask).

These findings highlight the challenge of generalizing across geographic domains, even when using advanced models. Although transfer learning improves metrics, it does not always translate into practically useful segmentations.

Cuadro 2: Performance comparison of U-Net models

Model	**Dice (Validation)**	**Dice (Real Images)**
Standard U-Net	0.9058	0.0278
Mini U-Net	0.8863	0.0278
U-Net + ResNet34	0.8653	**0.6620**

5. Conclusions

The results indicate that the standard U-Net architecture offers the highest precision on the original validation set, while the ResNet34-based model, thanks to transfer learning, shows better generalization to unseen environments, albeit with less defined segmentations. The Mini U-Net serves as an efficient alternative for scenarios with limited computational resources.

This work confirms the potential of semantic segmentation techniques for the automated analysis of photovoltaic deployment in urban settings. It also emphasizes the importance of combining quantitative metrics with qualitative assessments (e.g., using Grad-CAM), especially when evaluating domain adaptation.

Future work may focus on:

- Expanding the dataset with local images and more representative annotations.
- Refining the encoder with fine-tuning techniques tailored to southern European rooftops.
- Integrating complementary sources such as cadastral data, solar maps, or temporal sequences.
- Exploring more advanced architectures (e.g., vision Transformers or Swin-UNet).

Overall, this project lays the groundwork for intelligent tools for urban energy planning and large-scale solar self-consumption analysis.

6. References

- https://github.com/leticiacologan/TFM

ÍNDICE GENERAL

1. Introducción **1**
- 1.1. Estado del arte . . . 1
- 1.2. Motivación . . . 3
- 1.3. Objetivos . . . 4
- 1.4. Metodología . . . 4
- 1.5. Herramientas utilizadas . . . 6

2. Marco teórico **7**
- 2.1. Energía solar y contexto de implantación . . . 7
- 2.2. Inteligencia Artificial y su aplicación en energía . . . 8
- 2.3. Redes Neuronales . . . 8
  - 2.3.1. Redes Neuronales Convolucionales (CNN) . . . 11
  - 2.3.2. U-Net . . . 13
  - 2.3.3. Transfer learning . . . 15
  - 2.3.4. ResNet34 como Encoder Residual . . . 17

3. Adquisición y transformación de datos **20**
- 3.1. Dataset utilizado . . . 20
- 3.2. Filtrado previo mediante clustering . . . 21
  - 3.2.1. Extracción de características . . . 24
  - 3.2.2. Clustering con K-Means . . . 25
- 3.3. Data Augmentation . . . 28

4. Modelización y análisis de resultados **30**
- 4.1. Métricas de evaluación . . . 30
  - 4.1.1. Dice Coefficient . . . 30
  - 4.1.2. Intersection over Union (IoU) . . . 31
  - 4.1.3. Función de pérdida (Loss Function) . . . 32
  - 4.1.4. Entrenamiento de la CNN U-Net . . . 32
  - 4.1.5. Transfer learning de ResNet34 + U-Net . . . 41
  - 4.1.6. Análisis de resultados y comparación de modelos . . . 43

5. Evaluación del modelo **50**
5.1. Justificación de la elección del área de estudio 51
5.2. Obtención de resultados 52

6. Conclusiones **58**
6.1. Limitaciones 58
6.2. Líneas futuras 59

7. Planificación Temporal **62**

Bibliografía **63**

ÍNDICE DE FIGURAS

1. Predicción y mapa de activación del modelo **UNet** sobre dos ejemplos. III
2. Predicción de U-Net +ResNet34 sobre una imagen de Molina de Segura (imagen, máscara predicha). IV
3. Prediction and Grad-CAM activation maps from the **U-Net** model on two examples. VIII
4. Prediction from the U-Net + ResNet34 model on a real image (input, predicted mask). IX

2.1. Esquema de funcionamiento de la energía solar térmica en los hogares.[1] 7
2.2. Partes de una neurona.[2] 9
2.3. Estructura básica de una red neuronal.[3] 10
2.4. Estructura de una CNN.[4] 12
2.5. Estructura de una U-Net.[5] 15
2.6. Funcionamiento transfer learning.[6] 16
2.7. Arquitectura ResNet34.[7] 18
2.8. Ejemplo de jerarquía de clases en ImageNet utilizada durante el preentrenamiento de modelos como ResNet34. Fuente: [8]. 18

3.1. Código elaborado para la creación de máscaras 20
3.2. Imagen con su mascara binaria asociada 21
3.3. Imagen con su mascara binaria generada 21
3.4. Imagen de una casa con su mascara binaria asociada 22
3.5. Imagen sin paneles solares y su máscara binaria generada para representar la ausencia de instalaciones. 22
3.6. Imagen sin paneles solares y su máscara binaria generada para representar la ausencia de instalaciones. 23
3.7. Distribución de clases en el conjunto de imágenes seleccionadas. . . 23
3.8. Código elaborado para la extracción de features 24
3.9. Código elaborado para la fase final de extracción de features 24
3.10. Ejemplo visual del algoritmo KMeans.[9] 26
3.11. Código para aplicar K-means y posterior creación del dataframe . . 26
3.12. Clusters obtenidos tras aplicar K-Means 27

3.13. CSV generado con las anotaciones asociadas a parte de los clústers generados. 27
3.14. Visualización del primer clúster obtenido. 27
3.15. Visualización del segundo clúster obtenido. 28
3.16. Visualización del tercer clúster obtenido. 28

4.1. Fórmula y representación de Dice. [10] 31
4.2. Fórmula y representación de IoU. [10] 31
4.3. Imagen con su máscara y predicción de la misma asociada 32
4.4. Dataset personalizado para la carga de imágenes y máscaras binarias. 33
4.5. Inicialización del modelo U-Net 36
4.6. Inicialización del modelo Mini U-Net 38
4.7. Activación de *early stopping* tras 8 épocas sin mejora en la pérdida de validación. 40
4.8. Inicialización del modelo U-Net con encoder ResNet34 preentrenado en ImageNet. 42
4.9. Definición del optimizador Adam y la función de pérdida. 43
4.10. Evolución de las métricas de evaluación de Unet 44
4.11. Evolución de las métricas de evaluación de MiniUnet 44
4.12. Evolución de las métricas de evaluación de Unet + Resnet34 45
4.13. Predicción y mapa de activación del modelo **UNet** sobre dos ejemplos. 47
4.14. Predicción y mapa de activación del modelo **MiniUNet** sobre dos ejemplos. 48
4.15. Predicción y mapa de activación del modelo **Resnet34 + UNet** sobre dos ejemplos. 48

5.1. Ejemplo de imágenes obtenidas con su mascara asociada creada. . . 50
5.2. Interfaz Label Studio para la creación de las máscaras asociadas. . . 51
5.3. Localización de Molina de Segura. 51
5.4. Predicción de U-Net sobre una imagen de Molina de Segura (imagen, máscara predicha) 53
5.5. Predicción de U-Net sobre una imagen de Molina de Segura (imagen, máscara predicha y mapa Grad-CAM) 53
5.6. Predicción de Mini U-Net sobre una imagen de Molina de Segura (imagen, máscara predicha y mapa Grad-CAM). 54
5.7. Predicción de Mini U-Net sobre una imagen de Molina de Segura (imagen, máscara predicha). 54
5.8. Predicción de U-Net +ResNet34 sobre una imagen de Molina de Segura (imagen, máscara predicha). 55
5.9. Predicción de U-Net +ResNet34 sobre una imagen de Molina de Segura (imagen, máscara predicha y mapa Grad-CAM). 56

7.1. Diagrama de Gantt del proyecto. 62

ÍNDICE DE CUADROS

1. Comparativa de rendimiento de modelos U-Net IV
2. Performance comparison of U-Net models IX

1.1. Resumen de herramientas y tecnologías utilizadas en el proyecto. . . 6

4.1. Resumen de la arquitectura U-Net 34
4.2. Resumen de la arquitectura Mini U-Net 36
4.3. Comparativa detallada entre las arquitecturas U-Net y U-Net Mini. 39
4.4. Resumen de la arquitectura U-Net + codificador ResNet34 42
4.5. Comparativa de rendimiento entre modelos de segmentación 44
4.6. Comparativa de arquitecturas -Net (configuración y entrenamiento) 49
4.7. Comparativa de arquitecturas U-Net (ventajas y limitaciones) . . . 49

5.1. Resumen de evaluación de los modelos en Molina de Segura 57

ACRÓNIMOS

IA	Inteligencia Artificial
ML	Machine Learning (Aprendizaje Automático)
DL	Deep Learning (Aprendizaje Profundo)
CNN	Convolutional Neural Network (Red Neuronal Convolucional)
IoU	Intersection over Union (Intersección sobre Unión)
Dice	Dice Coefficient (Coeficiente de Dice)
BCE	Binary Cross-Entropy (Entropía Cruzada Binaria)
GPU	Graphics Processing Unit (Unidad de Procesamiento Gráfico)
UNet	U-Net (Red neuronal para segmentación basada en arquitectura encoder-decoder)
Grad-CAM	Gradient-weighted Class Activation Mapping (Mapeo de Activación de Clases con Gradientes)
BCEWithLogitsLoss	Binary Cross-Entropy with Logits Loss
ResNet	Residual Network (Red Residual)

1. INTRODUCCIÓN

La transición energética hacia fuentes renovables ha impulsado el desarrollo de soluciones tecnológicas innovadoras entre ellas el uso de Deep Learning, Big Data e imágenes satelitales para identificar oportunidades de instalación de placas solares en entornos urbanos y residenciales.

La energía solar, cada vez más accesible gracias a la reducción de costes y al impulso del autoconsumo, presenta un alto potencial de adopción, aunque todavía persisten desafíos para identificar y priorizar a los clientes potenciales.

Frente a la ineficiencia de los enfoques comerciales tradicionales, estas tecnologías permiten optimizar la captación de clientes, reducir costes y acelerar la adopción solar, lo que se traduce en estrategias de ventas más precisas y una posición fortalecida en el sector de las energías renovables. Este proyecto se desarrollará en colaboración con U4IMPACT para la energética portuguesa Galp.

1.1. Estado del arte

El desarrollo de las energías renovables ha motivado la incorporación de herramientas tecnológicas avanzadas en los procesos de planificación y análisis energético. Entre las distintas fuentes disponibles, la energía solar fotovoltaica ha experimentado un crecimiento significativo en entornos urbanos y residenciales, favorecida por la reducción de costes y la implementación de políticas de fomento al autoconsumo. Sin embargo, su adopción presenta una distribución geográfica heterogénea, condicionada por diversos factores técnicos, económicos y sociales.

Algunos aspectos como la irradiación solar disponible, la orientación e inclinación de las cubiertas, la capacidad de inversión de los hogares, la existencia de incentivos económicos, así como las percepciones socioculturales, inciden directamente en la viabilidad de cada proyecto fotovoltaico. En consecuencia, resulta muy importante contar con información precisa, actualizada y espacialmente detallada para evaluar adecuadamente el potencial de generación solar en distintas regiones y optimizar las estrategias de instalación de las placas.

Tradicionalmente, la recopilación de datos sobre instalaciones solares se ha llevado a cabo mediante métodos manuales o a través de registros administrativos, muchas veces incompletos o de acceso restringido. Esta aproximación conlleva importantes limitaciones en términos de cobertura, frecuencia de actualización y escalabilidad.

En este contexto, el uso de imágenes aéreas de alta resolución obtenidas a partir de satélites, junto con algoritmos de Deep Learning, ha emergido como una alternativa eficaz para automatizar la identificación y caracterización de instalaciones fotovoltaicas [11, 12].

Dentro de estas técnicas, la segmentación semántica se ha consolidado como una herramienta particularmente eficaz, al permitir la detección precisa de objetos en imágenes mediante la clasificación de cada píxel. Este enfoque resulta especialmente útil para el reconocimiento de paneles solares, cuya presencia puede pasar desapercibida en métodos de análisis más generales.

Modelos como **U-Net**, una arquitectura de redes neuronales convolucionales originalmente desarrollada para aplicaciones en imágenes médicas [13], han demostrado un rendimiento destacado en esta tarea gracias a su capacidad para combinar información de contexto global con detalles espaciales finos. Asimismo, se han desarrollado variantes versiones que incorporan modelos preentrenados como **ResNet34**, los cuales permiten reutilizar características aprendidas en tareas visuales genéricas, adaptándolas a contextos específicos.

El desempeño de estos modelos depende, en gran medida, de la calidad y representatividad de los datos utilizados en su entrenamiento. En este sentido, la disponibilidad creciente de conjuntos de datos abiertos ha facilitado el desarrollo y validación de soluciones de segmentación aplicables a distintos entornos geográficos. Estos datasets permiten el entrenamiento supervisado de los modelos, mejorando su capacidad de generalización y permitiendo la evaluación comparativa entre distintas aproximaciones.

Para este proyecto se utilizará el siguiente conjunto de datos: `https://www.nature.com/articles/s41597-023-01951-4` [14]. Como se describe en el artículo, las imágenes, de diferentes zonas de Francia, se obtuvieron a través de Google Earth Engine y el portal IGN Geoservices, y las anotaciones fueron realizadas por voluntarios mediante *crowdsourcing*.

No obstante, uno de los principales desafíos asociados al uso de modelos de aprendizaje profundo en tareas de segmentación geoespacial es el fenómeno conocido como domain shift. Este ocurre cuando un modelo entrenado con imágenes

de una determinada región o proveedor pierde capacidad predictiva al aplicarse en un contexto visual diferente, debido a variaciones en la resolución, condiciones de iluminación, ángulos de captura o estilos arquitectónicos predominantes. Esta problemática resalta la necesidad de diseñar modelos robustos y adaptativos, capaces de mantener su rendimiento frente a cambios sustanciales en las condiciones del dominio de aplicación.

Para mitigar estos efectos, técnicas como *transfer learning* y *fine-tuning* se han consolidado como metodologías efectivas. Estas estrategias permiten aprovechar redes previamente entrenadas sobre grandes corpus de imágenes generales como ImageNet, adaptándolas posteriormente a tareas específicas mediante el uso de conjuntos de datos más reducidos. De este modo, se optimiza el proceso de entrenamiento, se mejora la capacidad de generalización del modelo y se reducen los recursos computacionales necesarios.

En conjunto, la integración de imágenes aéreas de alta resolución, técnicas avanzadas de segmentación semántica y metodologías de aprendizaje profundo constituye una oportunidad sólida y eficaz para mejorar el conocimiento sobre la distribución y características de las instalaciones solares. Ello permite no solo facilitar la toma de decisiones en políticas energéticas y planificación urbana, sino también contribuir a la aceleración del proceso de transición hacia un modelo energético más sostenible, eficiente y basado en datos.

1.2. Motivación

El principal incentivo para la elección de este tema es poder entender cómo funciona la tecnología que hay detrás de la Inteligencia Artificial, en especial aquellas relacionadas con el Machine Learning y el Deep Learning.

Estas técnicas están revolucionando múltiples sectores, proporcionando nuevas formas de analizar datos, optimizar procesos y mejorar la toma de decisiones. El sector energético, y en particular las energías renovables, no es ajeano a esta transformación y cada vez son mas las soluciones basadas en IA que se toman para optimizar el aprovechamiento de los recursos disponibles.

Asimismo, la falta de experiencia previa del alumno en estas tecnologías convierte este proyecto en un desafío enriquecedor, que no solo permite adquirir competencias técnicas en uno de los campos con mayor proyección futura, sino que también proporciona una visión práctica sobre su aplicabilidad real en un sector estratégico como el de la energía solar.

1.3. Objetivos

El objetivo principal de este trabajo es desarrollar un sistema automático de segmentación semántica que permita detectar paneles solares en cubiertas de viviendas unifamiliares a partir de imágenes aéreas. Este sistema servirá como base para identificar zonas urbanas con instalaciones fotovoltaicas existentes y, en fases posteriores, podrá ampliarse para apoyar la estimación del potencial solar en áreas residenciales aún no equipadas.

Esta solución contribuirá a optimizar el proceso comercial de captación de cliente de la energética portuguesa Galp, permitiendo una selección más precisa y estratégica de los hogares más aptos para la transición energética hacia el autoconsumo solar. Al reducir la dependencia de enfoques tradicionales basados en análisis manuales, se busca lograr una disminución significativa de los costes de adquisición, una mejora en las tasas de conversión de ventas, y un fortalecimiento de la posición de la empresa en el mercado de las energías renovables.

Asimismo, este proyecto representa una oportunidad para demostrar cómo el uso del Deep Learning combinado con Big Data geoespacial puede acelerar la adopción de soluciones sostenibles a gran escala, contribuyendo a su vez a la transición energética global.

1.4. Metodología

En este apartado se explicarán las diferentes fases del proyecto que han llevado la obtención de los objetivos previamente establecidos.

Etapas seguidas para la elaboración del proyecto.

1. Introducción a los modelos de predicción de Deep Learning y redes neuronales.
2. Obtención y análisis del dataset
3. Clasificación previa de imágenes mediante clustering
4. Diseño y desarrollo del modelo de segmentación.
5. Entrenamiento y validación.
6. Análisis de resultados.
7. Conclusiones establecidas.

8. Establecimiento de líneas futuras.

Para comenzar con la elaboración de este proyecto, ha sido necesario invertir tiempo en familiarizarse con los modelos de Machine y Deep Learning y entender el funcionamiento que existe detrás de la estructura de cada uno de ellos junto con la influencia de sus parámetros.

Una vez asentados los conceptos mencionados, se procede a analizar y preparar el dataset. Se parte de un conjunto de imágenes aéreas de distintas tipologías de terrenos y edificaciones a las que será necesario hacer un preprocesamiento, estandarización de soluciones y generación de máscaras correspondientes que nos permitirán entrenar el modelo de segmentación.

Previo al entrenamiento de la red convolucional se aplicarán técnicas de Machine Learning para optimizar el proceso de segmentación. Mediante K-Means, técnica de clustering no supervisado, se distinguirá entre imágenes que representan viviendas y aquellas que no, como terrenos vacíos, bosques u otros elementos no relevantes. De esta manera, se hace un primer filtrado del dataset enfocando la segmentación unicamente en imágenes candidatas a contener instalaciones fotovoltaicas.

A continuación, sobre las imágenes ya clasificadas, se implementaron varios modelos de segmentación semántica basado en la arquitectura U-Net aplicando tambien técnicas de transfer learning para mejorar la generalización del modelo.

El modelo se entrenó utilizando las imágenes filtradas, empleando métricas especializadas como el Dice Coefficient y el IoU para monitorizar el aprendizaje y la capacidad de segmentación.

Se realizó un análisis cuantitativo y cualitativo de los resultados obtenidos, observando tanto las métricas numéricas como las predicciones visuales comparadas con las máscaras reales.

Como prueba de concepto, se seleccionó un conjunto de 25 imágenes aéreas del municipio de Molina de Segura (Murcia), recopiladas manualmente mediante la plataforma Google Earth Pro. Sobre este subconjunto, se aplicaron los distintos modelos de segmentación entrenados previamente para evaluar su capacidad de generalización en un entorno visual y geográfico distinto.

Aunque no se generó un ranking de viviendas priorizadas, esta evaluación representa un paso preliminar clave para validar la viabilidad técnica del enfoque, con vistas a su futura aplicación en estrategias comerciales de empresas como Galp.

Para terminar, se extraen las conclusiones en relación al desempeño de los modelos desarrollados, evaluando la capacidad de detección de instalaciones solares quedando abierta la posibilidad a futuro de añadir modelos adicionales que conduzcan a mejores resultados y nuevas métricas ad-hoc.

El codigo para la elaboración de este proyecto se puede consultar en el siguiente repositorio: https://github.com/leticiacologan/TFM

1.5. Herramientas utilizadas

En la Tabla 1.1 se presentan las principales herramientas y tecnologías utilizadas en el desarrollo del proyecto, clasificadas según su función principal dentro del flujo de trabajo.

Herramienta / Librería	**Uso principal**	**Tecnología / Función**
Python	Desarrollo completo	Lenguaje de programación
Scikit-learn	Clustering de imágenes (KMeans)	Machine Learning clásico
Google Maps API	Descarga de imágenes aéreas reales	API geoespacial
Pandas / Numpy	Gestión y preprocesamiento de datos	Librerías de datos
PIL / OpenCV	Manipulación básica de imágenes	Librerías de imágenes
PyTorch	Entrenamiento de redes neuronales	Framework Deep Learning
SMP (Segmentation Models PyTorch)	Implementación de U-Net con ResNet34	Segmentación avanzada
Matplotlib / Seaborn	Visualización de métricas y resultados	Librerías gráficas
tqdm	Barras de progreso	Soporte para iteraciones largas
Google Colab	Entrenamiento en GPU	Entorno de desarrollo cloud

Cuadro 1.1: Resumen de herramientas y tecnologías utilizadas en el proyecto.

2. MARCO TEÓRICO

En esta sección se desarrollan los conceptos fundamentales para comprender las distintas áreas de conocimiento que sustentan el proyecto, abordando desde los fundamentos de la energía solar y la Inteligencia Artificial, hasta técnicas específicas de segmentación de imágenes, redes neuronales convolucionales (CNN) y clustering, que permiten cumplir con el objetivo de este proyecto.

2.1. Energía solar y contexto de implantación

La energía solar fotovoltaica es una forma de energía renovable que permite la conversión directa de la radiación solar en electricidad mediante el uso de dispositivos denominados paneles solares. Esta conversión se realiza mediante el llamado efecto fotoeléctrico, un fenómeno físico mediante el cual determinados materiales semiconductores, al recibir fotones procedentes de la luz solar, liberan electrones que generan una corriente eléctrica continua. Se trata de una fuente de energía limpia, inagotable y no contaminante, siendo una alternativa a los combustibles [15].

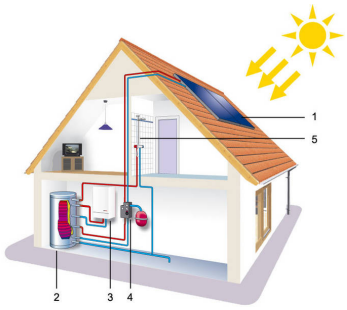

Figura 2.1: Esquema de funcionamiento de la energía solar térmica en los hogares.[1]

En los últimos años, la reducción progresiva del coste de los paneles solares, junto con la creciente conciencia ambiental y el impulso del auto consumo, ha

favorecido su adopción en el ámbito residencial. Sin embargo, la identificación de emplazamientos adecuados para la instalación de estos sigue siendo una tarea compleja, especialmente en zonas densamente construidas.

2.2. Inteligencia Artificial y su aplicación en energía

La Inteligencia Artificial (IA) permite a las máquinas aprender y tomar decisiones a partir de datos. En particular, el Machine Learning y su rama más avanzada, el Deep Learning, han demostrado un alto rendimiento en tareas complejas como el procesamiento de imágenes o computer vision. [16] [17].

En el ambito energético, estas tecnologías se están utilizando para predecir la generación solar, detectar fallos en instalaciones y optimizar el consumo energético [18].

Este proyecto aplica IA para identificar y priorizar viviendas con alto potencial fotovoltaico a partir de imágenes aéreas, contribuyendo así a la automatización de procesos que tradicionalmente eran manuales, costosos y lentos. Para compañías como Galp, esto supone una mejora en la eficiencia operativa y una aceleración en la transición hacia un modelo energético más sostenible.

2.3. Redes Neuronales

El objetivo de esta sección es dar una breve introducción a cerca de las redes neuronales ya que constituirán la base para cumplir con el objetivo de este proyecto.

Dentro del ámbito del Deep Learning, las redes neuronales se definen como modelos de predicción que buscan imitar el comportamiento del cerebro humano mediante la conexión entre neuronas.

En primer lugar, se explicará el concepto de neurona y su estructura con el fin de llegar a entender el funcionamiento de las mismas.

Concepto de neurona.
Desde el punto de vista científico, una neurona se define como un conjunto de células nerviosas capaz de recibir datos del entorno exterior al cuerpo, procesarlos y entregar una respuesta inteligente de acuerdo con el hecho percibido.

Las partes de la neurona que permiten llevar a cabo este proceso son las siguientes:

1. **Dendritas**: captan la información del exterior y la envían al cuerpo de la neurona.
2. **Cuerpo neuronal**: región más ancha donde tiene lugar el procesamiento de la información.
3. **Axón**: extremo de la neurona que guía la transmisión de las señales de respuesta en función a la información procesada.

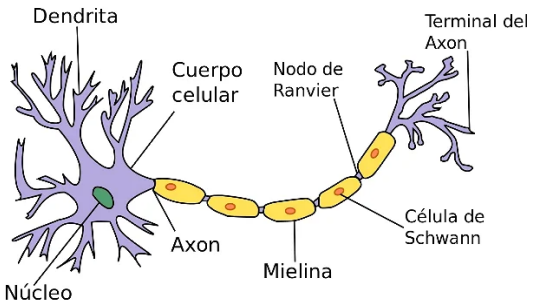

Figura 2.2: Partes de una neurona.[2]

Estructura y funcionamiento de una red neuronal.

Como se ha mencionado anteriormente, los algoritmos de redes neuronales buscan replicar, en términos computacionales, la estructura funcional del cerebro humano. Su objetivo es construir un modelo capaz de aprender automáticamente a partir de datos mediante la iteración de tareas, mejorando progresivamente su rendimiento.

Estas redes se componen de múltiples capas de procesamiento interconectadas. Se distinguen tres tipos fundamentales:

- **Capa de entrada (input layer)**: recibe los datos iniciales del problema.
- **Capas ocultas (hidden layers)**: realizan transformaciones intermedias aplicando funciones no lineales. Una red puede contener varias de estas capas, lo que permite modelar relaciones complejas entre los datos.
- **Capa de salida (output layer)**: genera la predicción final del modelo.

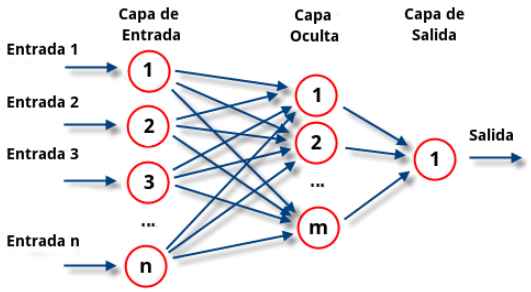

Figura 2.3: Estructura básica de una red neuronal.[3]

Durante el entrenamiento supervisado, las entradas se multiplican por los pesos de la red, se suman junto a un término de sesgo, y se aplica una función de activación. El resultado se propaga hacia la siguiente capa, y el proceso se repite.

Matemáticamente, una neurona puede representarse como:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

donde:

- x_i: entradas de la neurona.
- w_i: pesos asociados a cada entrada.
- b: término de sesgo (bias).
- f: función de activación.
- y: salida de la neurona.

El entrenamiento de la red se basa en ajustar los pesos utilizando algoritmos como el descenso del gradiente, con el objetivo de minimizar una función de pérdida definida según el problema.

Las funciones de activación más comunes incluyen:

- **Función sigmoide:**

$$f(x) = \frac{1}{1 + e^{-x}}$$

  Transforma los valores en un rango entre 0 y 1. Suele utilizarse en problemas de clasificación binaria.

- **Función ReLU (Rectified Linear Unit):**

$$f(x) = \text{máx}(0, x)$$

  Favorece el aprendizaje rápido y es muy utilizada en redes profundas.

- **Función tangente hiperbólica (tanh):**

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

  Similar a la sigmoide pero con salida entre -1 y 1.

- **Función Softmax:**

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \quad \text{para } i = 1, \ldots, n$$

  Convierte un vector de valores en una distribución de probabilidad. Es típica en la capa de salida de modelos de clasificación multiclase.

2.3.1. Redes Neuronales Convolucionales (CNN)

Las Redes Neuronales Convolucionales son una clase especializada de redes neuronales diseñadas para trabajar específicamente con datos con una estructura espacial, como las imágenes. A diferencia de las redes neuronales tradicionales, las CNN son capaces de capturar patrones locales dentro de una imagen como bordes, texturas o formas, lo cual hace que sean la mejor opción para tareas de computer vision. [17][19].

Arquitectura básica de una CNN.

Una CNN típica está compuesta por una secuencia de capas especializadas que transforman progresivamente la imagen de entrada en una representación de características más abstracta. Cada tipo de capa cumple una función específica en este proceso de transformación y aprendizaje jerárquico de la información visual, permitiendo que la red identifique patrones desde los más simples (como bordes o texturas) hasta los más complejos (como formas u objetos). A continuación, se describen las principales capas que conforman una CNN:

- **Capas convolucionales (Convolutional Layers):** aplican filtros sobre la imagen para extraer características locales como bordes, texturas o formas. Cada filtro genera un mapa de activación que destaca la presencia de una característica específica.

Un aspecto importante en estas capas es el *padding*, que consiste en añadir píxeles (generalmente con valor cero) alrededor de la imagen de entrada. Esto permite mantener constante la resolución espacial tras la convolución, evitar la pérdida de información en los bordes y facilitar que las capas más profundas tengan acceso a características periféricas.

- **Funciones de activación:** introducen no linealidad (usualmente ReLU) tras cada operación de convolución, permitiendo modelar relaciones complejas entre las características detectadas.
- **Capas de pooling (submuestreo):** reducen la resolución espacial de las representaciones, manteniendo la información más relevante y disminuyendo la complejidad computacional. La técnica más común es el *max pooling* que toma el valor máximo de una región.
- **Capas completamente conectadas (Fully Connected):** integran las características extraídas para producir la salida final, como una clasificación o un mapa de segmentación.
- **Capa de salida:** Utiliza funciones como softmax para tareas de clasificación multiclase.

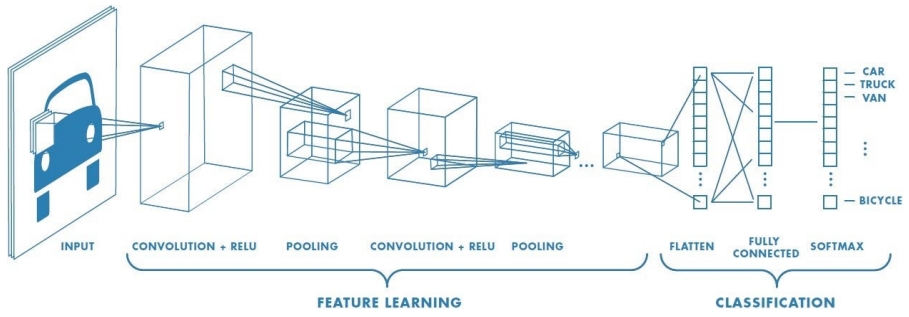

Figura 2.4: Estructura de una CNN.[4]

Ventajas de una CNN.

- **Extracción automática de características:** No es necesario definir manualmente qué patrones buscar; la red los aprende directamente de los datos.
- **Invarianza espacial parcial:** Pueden reconocer objetos incluso si están desplazados o rotados ligeramente.
- **Escalabilidad:** Se adaptan bien a grandes volúmenes de datos como imágenes satelitales de alta resolución.

Aplicación en segmentación semántica.

En este proyecto, en lugar de limitarse a clasificar imágenes completas, se emplea una CNN para llevar a cabo segmentación semántica, una técnica que asigna una etiqueta a cada píxel de la imagen. Este enfoque permite localizar con precisión las zonas ocupadas por paneles solares dentro de imágenes aéreas, lo que resulta fundamental para evaluar su presencia y distribución en entornos urbanos o residenciales.

A continuación, se describen en detalle los componentes clave de la arquitectura empleada en este proyecto: la red U-Net, utilizada para la segmentación semántica de imágenes; el encoder ResNet34, que actúa como extractor de características visuales profundas; y la técnica de transfer learning, fundamental para reutilizar conocimiento previamente aprendido y optimizar el entrenamiento en contextos específicos como el que aquí se aborda.

2.3.2. U-Net

Para abordar las tareas previamente citadas, se ha optado por una arquitectura especializada como U-Net, diseñada específicamente para segmentación. Su principal ventaja radica en su capacidad para combinar contexto global con información local detallada, lo que la convierte en una de las arquitecturas más efectivas para la detección precisa de estructuras en imágenes.

Esta red fue originalmente desarrollada para aplicaciones médicas, como la segmentación de órganos o tejidos en imágenes de resonancia magnética. Sin embargo, su versatilidad y alto rendimiento han favorecido su adopción en múltiples áreas, incluyendo la visión por computador en imágenes satelitales, como es el caso de este proyecto. [20]

Estructura de U-Net.

La arquitectura U-Net se compone de dos partes principales:

- **Bloque de contracción (encoder):** Esta sección es similar a una red convolucional clásica. Contiene capas de convolución seguidas por funciones de activación ReLU y operaciones de max pooling. El propósito es reducir progresivamente el tamaño espacial de la imagen mientras se aumenta la profundidad de las representaciones, capturando así información contextual de alto nivel. La operación de convolución se define como:

$$x^{(l)} = \sigma(W^{(l)} * x^{(l-1)} + b^{(l)})$$

donde $*$ representa la convolución, $W^{(l)}$ y $b^{(l)}$ son los pesos y sesgos de la capa l, y σ es la función de activación ReLU:

$$\sigma(z) = \text{máx}(0, z)$$

La operación de max pooling se puede expresar como:

$$x_{i,j}^{(l)} = \max_{(m,n) \in R(i,j)} x_{m,n}^{(l-1)}$$

- **Bloque de expansión (decoder):** A través de operaciones de upsampling o convoluciones transpuestas, esta parte incrementa la resolución espacial de las características extraídas. Lo más distintivo de U-Net es que, en cada etapa del decoder, se realiza una concatenación directa con las salidas correspondientes del encoder a través de conexiones tipo skip.

  La convolución transpuesta se define como:

  $$x^{(l)} = \sigma(W^{(l)} \star x^{(l-1)} + b^{(l)})$$

  donde $\star$ representa la convolución transpuesta.

  En cada etapa del decoder, se realiza una concatenación con la salida correspondiente del encoder:

  $$x^{(l)} = \text{concat}(x_{\text{decoder}}^{(l)}, x_{\text{encoder}}^{(l)})$$

  Esto permite recuperar detalles espaciales finos y mejorar la precisión de la segmentación. Estas conexiones permiten recuperar detalles espaciales finos que de otro modo se perderían durante la compresión, mejorando significativamente la precisión de la segmentación.

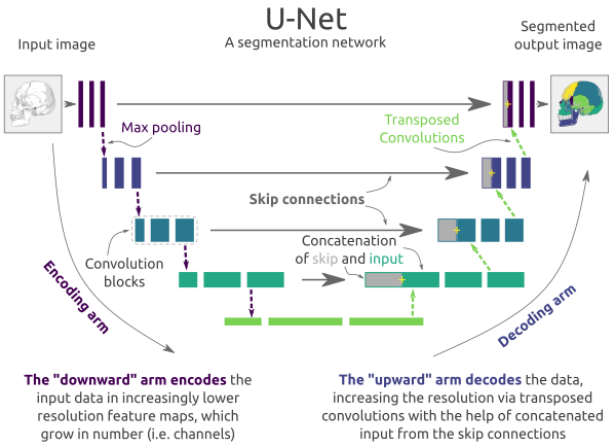

Figura 2.5: Estructura de una U-Net.[5]

Este diseño simétrico en forma de U permite que la red aprenda tanto la ubicación como la identidad de las regiones de interés. Gracias a esta estructura, U-Net es especialmente eficaz en situaciones donde los objetos a segmentar tienen formas irregulares, tamaños variados o aparecen sobre fondos complejos, como ocurre con los paneles solares en entornos urbanos.

Además, U-Net requiere relativamente pocos datos para lograr buenos resultados, lo cual es otra de las razones por las que se ha popularizado en tareas donde la disponibilidad de datos etiquetados es limitada.

2.3.3. Transfer learning

Previamente a explicar la arquitectura del encoder ResNet34, se desarrollará el concepto de transfer learning, una técnica fundamental dentro del Deep Learning que ha permitido mejorar significativamente el rendimiento de modelos en contextos donde los datos etiquetados son limitados o difíciles de obtener.

El transfer learning consiste en aprovechar el conocimiento adquirido por un modelo en una tarea previa para aplicarlo en una tarea diferente pero relacionada. Esta estrategia es especialmente útil cuando se dispone de pocos datos etiquetados para la tarea objetivo, ya que permite reducir tanto el tiempo de entrenamiento como la cantidad de datos necesarios.

En lugar de entrenar un modelo desde cero, lo cual puede ser computacionalmente costoso y requerir grandes volúmenes de datos, se parte de una red neuronal

previamente entrenada en una tarea general, y se adapta a una nueva tarea específica mediante técnicas como el *fine-tuning* o el uso del modelo como extractor de características.

Este proceso puede representarse formalmente como:

$$\mathcal{M}_{\text{target}} = \text{FineTune}(\mathcal{M}_{\text{source}}, \mathcal{D}_{\text{target}})$$

donde:

- $\mathcal{M}_{\text{source}}$ es el modelo preentrenado sobre el conjunto de datos fuente $\mathcal{D}_{\text{source}}$,
- $\mathcal{D}_{\text{target}}$ es el conjunto de datos de la tarea destino,
- $\mathcal{M}_{\text{target}}$ es el modelo adaptado a la nueva tarea,
- FineTune$(\cdot)$ representa el proceso de ajuste de los pesos a la tarea destino.

En términos simples, el modelo preentrenado actúa como una base de conocimiento visual capaz de reconocer patrones generales (como bordes, texturas o formas) que resultan útiles para múltiples tareas de visión por computador. A partir de esta base, se puede reentrenar únicamente la parte final del modelo, o ajustarlo completamente, para adaptarlo al nuevo conjunto de datos.

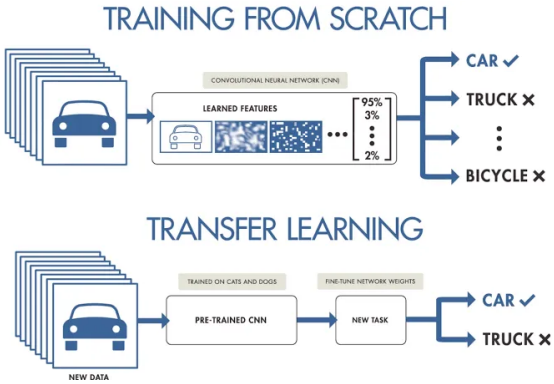

Figura 2.6: Funcionamiento transfer learning.[6]

Fine-tuning: Esta técnica consiste en tomar una red preentrenada y continuar su entrenamiento en el nuevo conjunto de datos, permitiendo actualizar parcialmente (o totalmente) sus pesos. Generalmente, las capas iniciales, que capturan características visuales muy generales, se mantienen congeladas, mientras que las capas más profundas , más específicas, se reajustan para adaptarse a la tarea final. Este enfoque permite conservar el conocimiento previamente adquirido mientras se ajusta a las particularidades del nuevo dominio, logrando un mejor equilibrio entre precisión y eficiencia computacional.

2.3.4. ResNet34 como Encoder Residual

ResNet34 es una red neuronal convolucional profunda que forma parte de la familia de arquitecturas **ResNet** (Residual Networks), introducidas por He et al. en 2015 [21]. Estas redes han supuesto un avance notable en el campo del aprendizaje profundo, al permitir entrenar modelos significativamente más profundos sin incurrir en los problemas clásicos de degradación del rendimiento o desaparición del gradiente.

El elemento clave de ResNet es el uso de las conexiones residuales o *skip connections*, que permiten que la entrada de un bloque se sume directamente a su salida. En lugar de aprender una función directa $H(\mathbf{x})$, se aprende una función residual $\mathcal{F}(\mathbf{x})$ tal que:

$$H(\mathbf{x}) = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}$$

donde:

- $\mathbf{x}$: entrada al bloque residual,
- $\mathcal{F}(\mathbf{x}, \{W_i\})$: función residual modelada por una o varias capas convolucionales con pesos W_i,
- $H(\mathbf{x})$: salida del bloque, como combinación aditiva entre la transformación aprendida y la entrada.

Este mecanismo mejora la retropropagación del gradiente, evitando que se atenúe al atravesar múltiples capas, y permite el entrenamiento de redes más profundas sin pérdida de rendimiento.

Matemáticamente, un bloque residual típico puede expresarse como:

$$\mathbf{y} = \sigma\left(\mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}\right)$$

donde σ representa una función de activación no lineal (como ReLU).

ResNet34 está compuesta por 34 capas organizadas en bloques residuales consecutivos. Cada bloque contiene dos capas convolucionales con filtros de 3×3, seguidas por operaciones de normalización por lotes (batch normalization) y activaciones ReLU. La salida de cada bloque se suma a su entrada para formar la salida residual. Esta profundidad permite a la red extraer representaciones jerárquicas altamente expresivas.

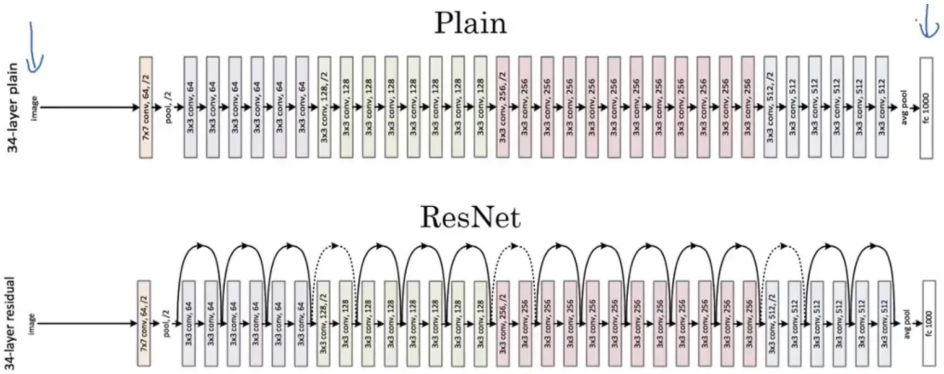

Figura 2.7: Arquitectura ResNet34.[7]

En este proyecto, ResNet34 se ha utilizado como encoder dentro de la arquitectura U-Net, aprovechando su capacidad para extraer características visuales discriminativas desde las primeras etapas del procesamiento. Concretamente, se emplea el tronco convolucional de ResNet34, eliminando la capa de clasificación final, y conectándolo al decodificador de U-Net mediante conexiones de salto.

Para maximizar el rendimiento y acelerar el entrenamiento, se ha empleado una versión preentrenada de ResNet34 en el conjunto **ImageNet**. Esta técnica, conocida como *transfer learning*, reutiliza los pesos $W_{\text{pretrained}}$ aprendidos a partir de millones de imágenes genéricas como punto de partida para tareas específicas:

$$W_{\text{pretrained}}^{\text{ImageNet}} \rightarrow W_{\text{iniciales}}^{\text{segmentación}}$$

Gracias a esta estrategia, se mejora la capacidad de generalización del modelo y se reduce la necesidad de grandes volúmenes de datos etiquetados.

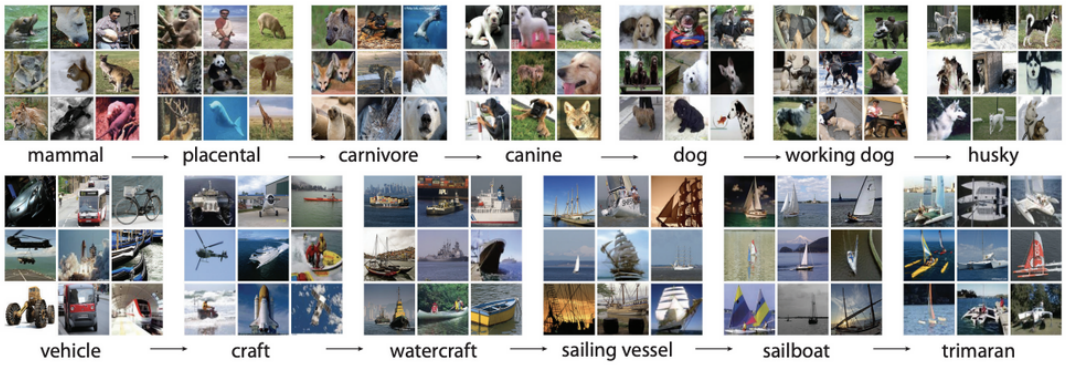

Figura 2.8: Ejemplo de jerarquía de clases en ImageNet utilizada durante el preentrenamiento de modelos como ResNet34. Fuente: [8].

La incorporación de ResNet34 como backbone en U-Net permite:

- Aprovechar características preentrenadas ricas en semántica visual.
- Acelerar el entrenamiento reduciendo el número de parámetros que deben aprenderse desde cero.
- Mejorar la segmentación, especialmente en escenarios con datos escasos o complejidad visual elevada.

En conjunto, esta estrategia híbrida combina lo mejor de dos enfoques: la expresividad de redes profundas residuales y la precisión espacial de U-Net en tareas de segmentación densa.

3. ADQUISICIÓN Y TRANSFORMACIÓN DE DATOS

Para el desarrollo del modelo propuesto, uno de los primeros pasos clave ha sido la recopilación, análisis y preparación del conjunto de datos. Dado que el rendimiento de cualquier sistema basado en aprendizaje automático depende en gran medida de la calidad de los datos de entrada, esta fase inicial es esencial para asegurar la robustez y generalización del modelo.

3.1. Dataset utilizado

Para la fase de entrenamiento del modelo se ha utilizado un conjunto de datos público proveniente de la plataforma Zenodo, disponible en el siguiente enlace: `https://zenodo.org/records/7358126`

Este dataset contiene imágenes aéreas de tejados, muchas de las cuales incluyen instalaciones fotovoltaicas. Cada imagen está asociada, en los casos positivos, a una máscara binaria que identifica la ubicación de los paneles solares.

Para aquellas imágenes que no contienen paneles, no se proporciona una máscara explícita, por lo que se ha generado una **máscara negra sintética** (sin anotaciones), indicando la ausencia total de instalaciones solares. Esto ha permitido mantener un formato unificado y coherente para todo el conjunto de datos.

```python
# Creamos las máscaras negras para imágenes sin máscara
for img_file in images_folder:
    # Asumimos que las máscaras tienen el mismo nombre que las imágenes
    mask_name = img_file.replace(".jpg", ".png")

    if mask_name not in masks_folder:
        # Cargar la imagen para saber su tamaño
        img_path = os.path.join(images_folder, img_file)
        img = Image.open(img_path)
        width, height = img.size

        # Crear una máscara negra
        empty_mask = np.zeros((height, width), dtype=np.uint8)
        empty_mask_img = Image.fromarray(empty_mask)

        # Guardar la máscara negra
        mask_path = os.path.join(masks_folder, mask_name)
        empty_mask_img.save(mask_path)

print("✅ Máscaras negras creadas para todas las imágenes sin máscara.")
```

```
✅ Máscaras negras creadas para todas las imágenes sin máscara.
```

Figura 3.1: Código elaborado para la creación de máscaras

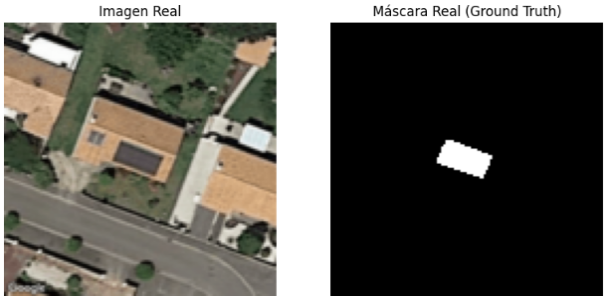

Figura 3.2: Imagen con su mascara binaria asociada

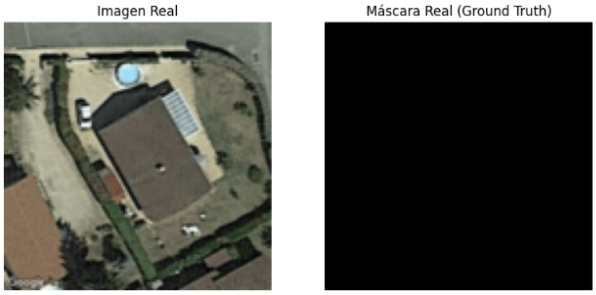

Figura 3.3: Imagen con su mascara binaria generada

3.2. Filtrado previo mediante clustering

Dado que el objetivo principal del sistema es segmentar paneles solares instalados específicamente en viviendas, es fundamental asegurar que el conjunto de entrenamiento esté compuesto por imágenes relevantes. Muchas de las muestras contenidas en el dataset original incluyen terrenos vacíos, zonas agrícolas, carreteras o áreas industriales que no aportan información útil para la tarea final.

Para evitar que el modelo aprenda patrones visuales irrelevantes y reducir el ruido durante el entrenamiento, se ha incorporado un paso de filtrado automático de imágenes, orientado a descartar aquellas que no contienen construcciones residenciales.

Este proceso se ha abordado en dos etapas consecutivas:

- **Extracción de características visuales:** se obtienen representaciones vectoriales de las imágenes, capturando patrones relevantes como texturas, bordes y formas.

- **Clustering no supervisado:** a partir de estas representaciones, se agrupan las muestras según su similitud visual, sin necesidad de contar con etiquetas previas, permitiendo organizar el conjunto de imágenes de forma estructurada.

Cabe destacar que las imágenes utilizadas en este proceso son a **color (RGB)**, lo que permite capturar mayor riqueza de información visual, como sombras, texturas y contrastes que serían difíciles de distinguir en imágenes en escala de grises. Esta representación más realista favorece una mejor extracción de características y una segmentación más precisa.

A continuación, se muestran ejemplos de imágenes del dataset junto con sus máscaras binarizadas, ilustrando algunos casos descartados por no contener construcciones residenciales relevantes.

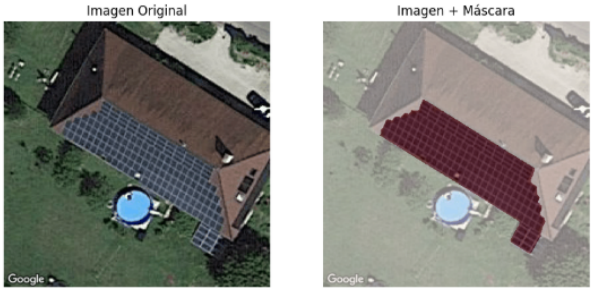

Figura 3.4: Imagen de una casa con su mascara binaria asociada

Figura 3.5: Imagen sin paneles solares y su máscara binaria generada para representar la ausencia de instalaciones.

Figura 3.6: Imagen sin paneles solares y su máscara binaria generada para representar la ausencia de instalaciones.

Este proceso de filtrado ha resultado especialmente útil para eliminar imágenes con escasa relevancia contextual y asegurar que el modelo se entrene únicamente con información útil.

Inicialmente, se disponía de un conjunto compuesto por **28.827 imágenes** cargadas en Google Colab. Tras aplicar el proceso de extracción de características y el posterior clustering, el conjunto final quedó reducido a **21.342 imágenes seleccionadas**, lo que supone una reducción del **26 %** del total.

Además, se realizó un análisis del balanceo de clases dentro del conjunto seleccionado. Tal como se observa en la Figura 3.7, aproximadamente el **60 %** de las imágenes contienen paneles solares, mientras que el **40 %** restante no los incluye. Esta proporción se considera razonablemente equilibrada para una tarea de segmentación binaria, ya que permite al modelo aprender de forma adecuada ambos contextos sin introducir un sesgo significativo hacia una clase dominante.

```
def calculate_class_balance(dataset):
    has_panel = 0
    no_panel = 0

    for i in tqdm(range(len(dataset))):
        _, mask = dataset[i]
        if mask.sum() > 0:
            has_panel += 1
        else:
            no_panel += 1

    total = has_panel + no_panel
    print(f"Total imágenes: {total}")
    print(f"Con paneles solares: {has_panel} ({(has_panel/total)*100:.2f}%)")
    print(f"Sin paneles solares: {no_panel} ({(no_panel/total)*100:.2f}%)")

# ✅ USO:

calculate_class_balance(dataset)
```

```
100%|██████████| 21342/21342 [4:24:22<00:00,  1.35it/s]Total imágenes: 21342
Con paneles solares: 12885 (60.37%)
Sin paneles solares: 8457 (39.63%)
```

Figura 3.7: Distribución de clases en el conjunto de imágenes seleccionadas.

Esta depuración mejora la calidad del conjunto de datos y contribuye a un entrenamiento más eficaz y preciso del modelo de segmentación.

3.2.1. Extracción de características

Con el fin de representar cada imagen del conjunto de datos de forma numérica y consistente, se ha empleado una red neuronal convolucional **ResNet50** previamente entrenada en el conjunto de datos ImageNet. Esta red, ampliamente utilizada en tareas de visión por computador, ha sido adaptada para funcionar como extractor de características visuales eliminando su capa de salida encargada de la clasificación.

```python
# 3) Prepara el extractor de features (ResNet50 sin la cabeza final)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
resnet = models.resnet50(pretrained=True).to(device).eval()
feat_extractor = torch.nn.Sequential(*list(resnet.children())[:-1])
```

```python
# 4) Transformaciones para extraer embeddings
tf = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize([0.485,0.456,0.406],[0.229,0.224,0.225]),
])
```

Figura 3.8: Código elaborado para la extracción de features

Al aplicar esta red a cada imagen, se obtiene un vector de **2048 dimensiones** que encapsula la información visual más relevante de la imagen, incluyendo patrones de textura, estructura y forma. Estos vectores permiten transformar el contenido visual en una representación compacta y de alto nivel que puede ser utilizada posteriormente como entrada para algoritmos de agrupamiento o análisis.

```python
# Extracción de features
paths, feats = [], []
for fn in tqdm(os.listdir(images_folder), desc="Extrayendo features"):
    if not fn.lower().endswith(('.jpg', 'jpeg', 'png', 'bmp', 'tif', 'tiff')):
        continue
    fp = os.path.join(images_folder, fn)
    img = Image.open(fp).convert('RGB')
    x = tf(img).unsqueeze(0).to(device)
    with torch.no_grad():
        v = feat_extractor(x).squeeze().cpu().numpy()
    feats.append(v.reshape(-1))
    paths.append(fn)

feats = np.vstack(feats)

# Guardar features y rutas
np.savez("/content/drive/MyDrive/bdappv/google/google/features_resnet50.npz", feats=feats, paths=np.array(paths))
print("✅ Features guardados como .npz")
```

```
Extrayendo features: 100%|██████████| 28827/28827 [4:07:43<00:00,  1.94it/s]
✅ Features guardados como .npz
```

Figura 3.9: Código elaborado para la fase final de extracción de features

Este tipo de representación facilita la comparación entre imágenes en un espacio semántico común, siendo especialmente útil para tareas posteriores donde se

requiere evaluar similitudes visuales o estructurar el conjunto de datos en función de sus características compartidas.

3.2.2. Clustering con K-Means

Una vez extraídas las representaciones numéricas de cada imagen a través del extractor de características, se ha aplicado el algoritmo de **K-Means** con el objetivo de agrupar automáticamente las imágenes en función de su similitud visual.

Este método de clustering no supervisado es ampliamente utilizado en tareas de análisis exploratorio y reducción de ruido, especialmente cuando se desconoce a priori la distribución de clases en el conjunto de datos.

El algoritmo K-Means tiene como objetivo minimizar la suma de distancias cuadráticas entre cada punto y su centroide asignado. Formalmente, se busca minimizar la siguiente función de coste:

$$J = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2 \tag{3.1}$$

donde:

- k es el número de clústeres definidos,
- C_i representa el conjunto de puntos asignados al clúster i,
- x_j es un vector de características perteneciente a C_i,
- μ_i es el centroide del clúster i.

El proceso de clustering se realiza de forma iterativa en los siguientes pasos:

1. Inicialización de k centroides $\mu_1, \mu_2, ..., \mu_k$, seleccionados aleatoriamente.
2. Asignación de cada punto x_j al clúster más cercano, es decir:

$$\text{cluster}(x_j) = \arg\min_i \|x_j - \mu_i\| \tag{3.2}$$

3. Recalcular los centroides como la media de los puntos asignados:

$$\mu_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j \tag{3.3}$$

4. Repetir los pasos 2 y 3 hasta que las asignaciones no varíen significativamente (convergencia).

La elección del valor de k (número de clústeres) se ha realizado de forma empírica, buscando un compromiso entre granularidad y representatividad. Una vez ejecutado el algoritmo, cada imagen ha quedado asignada a un clúster específico, almacenando esta información en un `DataFrame` para facilitar su posterior análisis.

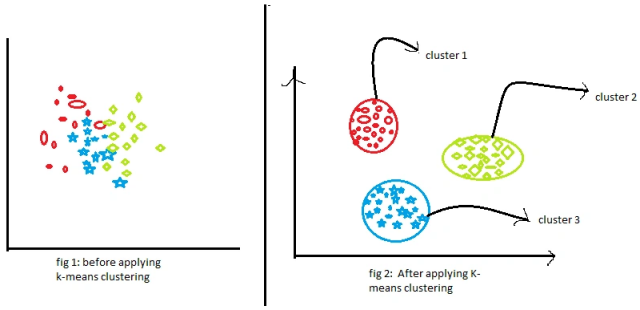

Figura 3.10: Ejemplo visual del algoritmo KMeans.[9]

Este agrupamiento ha permitido identificar patrones visuales comunes sin supervisión, diferenciando eficazmente entre imágenes relevantes (tejados residenciales) y no relevantes (zonas sin construcciones).

```
# 6) Aplica K-Means y crea un DataFrame con la asignación de clúster
km = KMeans(n_clusters=n_clusters, random_state=0).fit(feats)
df = pd.DataFrame({
    'image': paths,
    'cluster': km.labels_
})
```

Figura 3.11: Código para aplicar K-means y posterior creación del dataframe

```python
# 7) Muestra por consola ejemplos de cada clúster
for c in range(n_clusters):
    sample = df[df.cluster==c].sample(samples_per_cluster, random_state=0)['image'].tolist()
    print(f"\nCluster {c}  (total={len(df[df.cluster==c])} imágenes):")
    for img in sample:
        print("   ", img)
```

```
Cluster 0  (total=225 imágenes):
    JJBNX438CQBOFA.png
    PZUNK52A4XDHVW.png
    BEXOR7D57ZJLDW.png
    JEULL6FCDQNRYE.png
    NGIGF25A4KWMBF.png
    JCFJA4104QVWPG.png
    HPRPM67D0WHRNR.png
    GANDD71C3NUXFW.png
    HYNHNB75DSTFU.png
    UPJUF21C2LAPFG.png

Cluster 1  (total=1091 imágenes):
    UTTXZ6562IYMKB.png
    XPRMU1107TMPQN.png
    BDEUZ3FECRVSQZ.png
```

Figura 3.12: Clusters obtenidos tras aplicar K-Means

A partir de aquí, se procedió a una etiquetación manual de clústeres, asignando un valor binario a cada grupo (1 si contenía imágenes de viviendas, 0 en caso contrario).

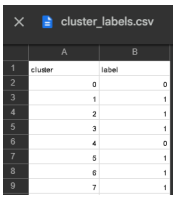

Figura 3.13: CSV generado con las anotaciones asociadas a parte de los clústers generados.

Para llevar a cabo esta tarea, se inspeccionó visualmente una muestra de imágenes representativas de cada clúster, como se muestra a continuación:

Figura 3.14: Visualización del primer clúster obtenido.

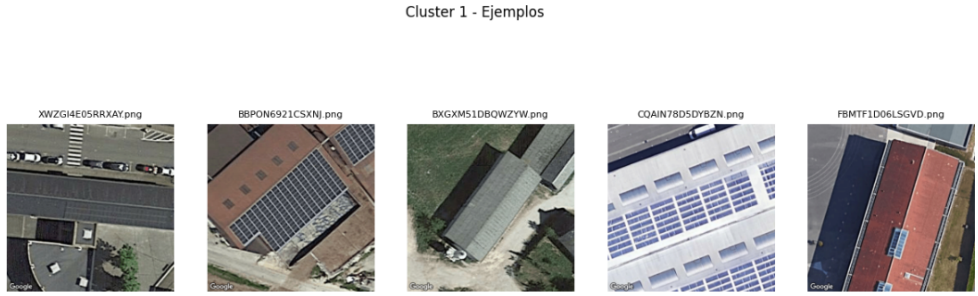

Figura 3.15: Visualización del segundo clúster obtenido.

Figura 3.16: Visualización del tercer clúster obtenido.

Finalmente, se fusionaron estas etiquetas manuales con el conjunto original de imágenes, lo que permitió filtrar automáticamente aquellas relevantes para el entrenamiento del modelo de segmentación. Este paso resultó clave para garantizar que la red neuronal se entrenara exclusivamente con datos representativos del entorno urbano residencial objeto de estudio.

3.3. Data Augmentation

En tareas de segmentación semántica, disponer de un conjunto de datos amplio y diverso es fundamental para mejorar la capacidad de generalización del modelo. Una estrategia habitual cuando se cuenta con un número reducido de imágenes etiquetadas consiste en emplear técnicas de *data augmentation*, que permiten aumentar artificialmente el tamaño y la variabilidad del conjunto de entrenamiento aplicando transformaciones a las imágenes originales.

El aumento de datos consiste en aplicar transformaciones geométricas o fotométricas a las imágenes originales con el objetivo de generar nuevas versiones sintéticas que preservan la información semántica, pero introducen variaciones que

ayudan al modelo a aprender de forma más robusta. Entre las transformaciones aplicadas comúnmente se encuentran:

- Rotaciones aleatorias
- Variaciones de brillo y contraste
- Giros horizontales o verticales
- Zoom o recortes parciales

Es importante destacar que, en el contexto de la segmentación semántica, toda transformación aplicada a una imagen debe aplicarse de forma idéntica a su máscara asociada. Esto asegura que la correspondencia entre los píxeles de la imagen y sus etiquetas se mantenga intacta tras la transformación.

No obstante, en este proyecto se dispone de un conjunto de datos suficientemente amplio y equilibrado, compuesto por **21.342 imágenes con sus correspondientes máscaras binarizadas**, lo que reduce significativamente el riesgo de sobreajuste y proporciona una buena cobertura de escenarios visuales variados.

Por esta razón, no se ha considerado necesario aplicar técnicas de aumento de datos, aunque su uso sigue siendo una estrategia válida y recomendable en casos donde el volumen de datos sea limitado o exista un fuerte desbalance entre clases.

4. MODELIZACIÓN Y ANÁLISIS DE RESULTADOS

En este apartado se mostrarán los pasos que se han seguido para la construcción final del modelo.

4.1. Métricas de evaluación

Para evaluar el rendimiento del modelo de segmentación semántica, se han empleado dos métricas ampliamente utilizadas en tareas de segmentación médica e imágenes aéreas:

- **Coeficiente de Dice**.
- **IoU (Intersection over Union)**

Ambas métricas cuantifican el grado de solapamiento entre la máscara predicha por el modelo y la máscara real (ground truth), siendo más apropiadas que una simple precisión global cuando se trata de evaluar áreas segmentadas.

En este proyecto no se ha establecido un umbral objetivo específico para dichas métricas, ya que el propósito principal es explorar el potencial del modelo para realizar segmentaciones precisas y consistentes. Por ello, se han considerado diversas arquitecturas y enfoques con el fin de analizar comparativamente su comportamiento y rendimiento.

4.1.1. Dice Coefficient

El coeficiente de Dice, también conocido como *F1 score para segmentación*, es una métrica que mide la similitud entre dos conjuntos de píxeles. Su fórmula es la siguiente:

$$\text{Dice} = \frac{2 \cdot |A \cap B|}{|A| + |B|} \tag{4.1}$$

donde A es el conjunto de píxeles predichos como positivos por el modelo, y B es el conjunto de píxeles realmente positivos según la máscara real.

Esta métrica toma valores entre 0 y 1, donde 1 representa una superposición perfecta entre las dos máscaras. El Dice es especialmente útil en situaciones donde las clases están desbalanceadas, como suele ocurrir en la segmentación de objetos pequeños (por ejemplo, paneles solares dentro de imágenes aéreas amplias).

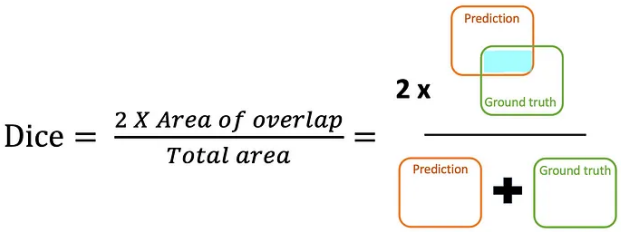

Figura 4.1: Fórmula y representación de Dice. [10]

4.1.2. Intersection over Union (IoU)

La métrica de Intersection over Union (IoU), también conocida como Jaccard Index, calcula la proporción entre la intersección y la unión de los píxeles predichos y reales. Se define como:

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \tag{4.2}$$

Al igual que el Dice, la IoU oscila entre 0 y 1, y se considera una métrica más estricta, ya que penaliza con mayor severidad las falsas predicciones (tanto falsos positivos como falsos negativos).[10]

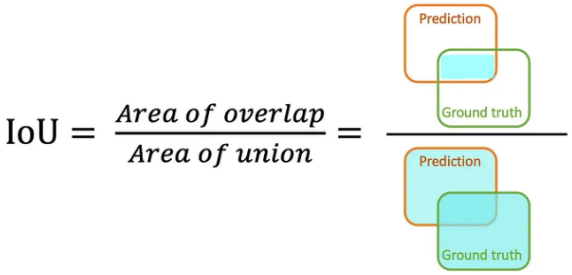

Figura 4.2: Fórmula y representación de IoU. [10]

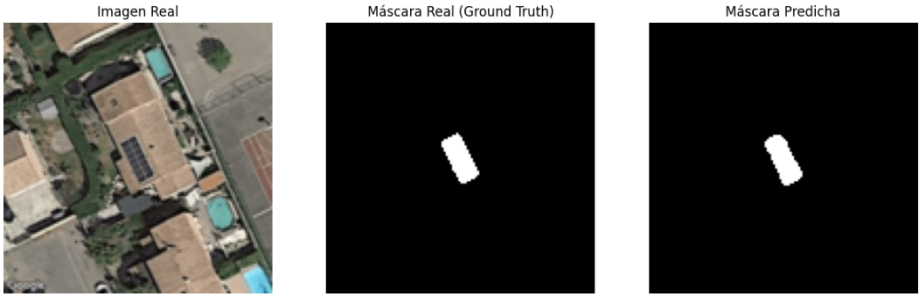

Figura 4.3: Imagen con su máscara y predicción de la misma asociada

4.1.3. Función de pérdida (Loss Function)

Para el entrenamiento del modelo, se ha empleado la función de pérdida *Binary Cross Entropy Loss* (BCELoss), definida como:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}[g_i \cdot \log(p_i) + (1 - g_i) \cdot \log(1 - p_i)]$$

donde p_i representa el valor predicho por el modelo (posterior a la activación sigmoide) para el píxel i, y g_i es el valor real correspondiente en la máscara binaria (ground truth). Esta función mide la discrepancia entre las probabilidades predichas y las etiquetas reales, penalizando fuertemente las predicciones incorrectas. [22]

La activación sigmoide se aplica a la salida del modelo para transformar los valores continuos en un rango de probabilidad entre 0 y 1, lo cual es esencial para que la función BCE interprete correctamente las predicciones como probabilidades de pertenencia a la clase positiva (por ejemplo, presencia de panel solar). [23]

La elección de esta pérdida responde al objetivo de maximizar la coincidencia espacial entre las regiones predichas y las verdaderas, enfocando el aprendizaje en las áreas relevantes del objeto de interés, incluso cuando estas ocupan una proporción reducida de la imagen.

4.1.4. Entrenamiento de la CNN U-Net

El proceso de entrenamiento de la red neuronal convolucional tipo U-Net se ha diseñado para abordar la tarea de segmentación semántica de paneles solares en imágenes aéreas. Este proceso se estructura en varias fases, desde la preparación de

los datos hasta la validación del modelo entrenado, garantizando así un desarrollo riguroso y una evaluación objetiva del rendimiento.

En primer lugar, se construyó un conjunto de datos personalizado compuesto por imágenes aéreas y sus correspondientes máscaras binarias, que indican la presencia o ausencia de paneles solares en cada píxel. Para homogeneizar la entrada al modelo y optimizar la eficiencia computacional, todas las imágenes fueron redimensionadas a una resolución fija y normalizadas. La máscara, por su parte, fue adaptada al mismo tamaño y convertida en una matriz binaria compatible con el modelo.

```python
#Dataset personalizado
class SolarPanelDataset(Dataset):
    def __init__(self, images_folder, masks_folder, transform=None):
        self.images_folder = images_folder
        self.masks_folder = masks_folder
        self.image_files = os.listdir(images_folder)
        self.transform = transform

    def __len__(self):
        return len(self.image_files)

    def __getitem__(self, idx):
        img_name = self.image_files[idx]
        img_path = os.path.join(self.images_folder, img_name)
        mask_name = img_name.replace(".jpg", ".png")
        mask_path = os.path.join(self.masks_folder, mask_name)

        image = Image.open(img_path).convert('RGB')
        mask = Image.open(mask_path).convert('L')

        if self.transform:
            image = self.transform(image)
            mask = self.transform(mask)

        mask = (mask > 0).float()

        return image, mask
```

Figura 4.4: Dataset personalizado para la carga de imágenes y máscaras binarias.

El conjunto completo se dividió en dos subconjuntos: entrenamiento (80 %) y validación (20 %). Esta separación garantiza que la evaluación del modelo se realice sobre datos no vistos, permitiendo medir de forma objetiva su capacidad de generalización.

La arquitectura utilizada se basa en el modelo U-Net, ampliamente reconocido por su eficacia en tareas de segmentación. Este tipo de red combina una estructura en forma de codificador-decodificador, en la que se reduce progresivamente la resolución espacial para extraer representaciones abstractas (codificación), y posteriormente se recupera la resolución original para producir una segmentación densa (decodificación). Las conexiones directas entre capas simétricas del codificador y el decodificador permiten conservar información espacial detallada, mejorando la precisión de los contornos segmentados.

En el presente estudio se han considerado dos variantes de esta arquitectura:

- **U-Net completa**: una implementación estándar con cuatro bloques de codificación y decodificación, canales iniciales de 64, y una capa intermedia de 1024 filtros. Esta arquitectura tiene mayor capacidad expresiva y está diseñada para aprender representaciones detalladas y complejas.

Cuadro 4.1: Resumen de la arquitectura U-Net

Componente	**Salida**	**Parámetros**
Capas de convolución (encoder + decoder)	Varios bloques CBR	29.9M
Capas de transposición (upsampling)	4 capas ConvTranspose2d	2.8M
Capa final	Conv2d 1×1 + sigmoid	65
Total de parámetros		31.04 millones
Parámetros entrenables		31.04 millones
Parámetros no entrenables		0
Tamaño de entrada	$128 \times 128 \times 3$	
Tamaño forward/backward	201.88 MB	
Tamaño del modelo (params)	118.42 MB	
Tamaño estimado total	320.48 MB	

Listing 4.1: Definición de la arquitectura `U_Net` estándar para segmentación

```
class UNet(nn.Module):
    def __init__(self):
        super(UNet, self).__init__()

        def CBR(in_channels, out_channels):
            return nn.Sequential(
                nn.Conv2d(in_channels, out_channels, 3,
                          padding=1),
                nn.BatchNorm2d(out_channels),
                nn.ReLU(inplace=True)
            )

        self.enc1 = nn.Sequential(CBR(3, 64), CBR(64, 64))
```

```
        self.enc2 = nn.Sequential(CBR(64, 128), CBR(128, 128)
                )
        self.enc3 = nn.Sequential(CBR(128, 256), CBR(256,
                256))
        self.enc4 = nn.Sequential(CBR(256, 512), CBR(512,
                512))

        self.pool = nn.MaxPool2d(2)

        self.middle = nn.Sequential(CBR(512, 1024), CBR(1024,
                 1024))

        self.up4 = nn.ConvTranspose2d(1024, 512, 2, stride=2)
        self.dec4 = nn.Sequential(CBR(1024, 512), CBR(512,
                512))

        self.up3 = nn.ConvTranspose2d(512, 256, 2, stride=2)
        self.dec3 = nn.Sequential(CBR(512, 256), CBR(256,
                256))

        self.up2 = nn.ConvTranspose2d(256, 128, 2, stride=2)
        self.dec2 = nn.Sequential(CBR(256, 128), CBR(128,
                128))

        self.up1 = nn.ConvTranspose2d(128, 64, 2, stride=2)
        self.dec1 = nn.Sequential(CBR(128, 64), CBR(64, 64))

        self.final = nn.Conv2d(64, 1, 1)

    def forward(self, x):
        e1 = self.enc1(x)
        e2 = self.enc2(self.pool(e1))
        e3 = self.enc3(self.pool(e2))
        e4 = self.enc4(self.pool(e3))

        m = self.middle(self.pool(e4))

        d4 = self.up4(m)
        d4 = torch.cat([d4, e4], dim=1)
        d4 = self.dec4(d4)

        d3 = self.up3(d4)
        d3 = torch.cat([d3, e3], dim=1)
        d3 = self.dec3(d3)

        d2 = self.up2(d3)
        d2 = torch.cat([d2, e2], dim=1)
        d2 = self.dec2(d2)
```

```
        d1 = self.up1(d2)
        d1 = torch.cat([d1, e1], dim=1)
        d1 = self.dec1(d1)

        out = torch.sigmoid(self.final(d1))
        return out
```

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = UNet().to(device)

optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)
criterion = nn.BCELoss()
```

Figura 4.5: Inicialización del modelo U-Net .

- **U-Net Mini**: una versión reducida con canales iniciales de 32 y un cuello de botella de 512 filtros, pensada para entornos con recursos computacionales limitados. A pesar de su menor capacidad, presenta un buen compromiso entre precisión y eficiencia.

Cuadro 4.2: Resumen de la arquitectura Mini U-Net

Componente	**Salida**	**Parámetros**
Capas de convolución (encoder + decoder)	Varios bloques CBR	6.5M
Capas de transposición (upsampling)	4 capas ConvTranspose2d	1.2M
Capa final	Conv2d 1×1 + sigmoid	33
Total de parámetros		7.77 millones
Parámetros entrenables		7.77 millones
Parámetros no entrenables		0
Tamaño de entrada	$128 \times 128 \times 3$	
Tamaño forward/backward	101.00 MB	
Tamaño del modelo (params)	29.62 MB	
Tamaño estimado total	130.81 MB	

Listing 4.2: Definición de la arquitectura `U_NetMini` utilizada para segmentación

```
class UNetMini(nn.Module):
    def __init__(self):
        super(UNetMini, self).__init__()

        def CBR(in_channels, out_channels):
            return nn.Sequential(
                nn.Conv2d(in_channels, out_channels, 3,
                          padding=1),
                nn.BatchNorm2d(out_channels),
                nn.ReLU(inplace=True)
            )

        self.enc1 = nn.Sequential(CBR(3, 32), CBR(32, 32))
        self.enc2 = nn.Sequential(CBR(32, 64), CBR(64, 64))
        self.enc3 = nn.Sequential(CBR(64, 128), CBR(128, 128)
                )
        self.enc4 = nn.Sequential(CBR(128, 256), CBR(256,
                256))

        self.pool = nn.MaxPool2d(2)

        self.middle = nn.Sequential(CBR(256, 512), CBR(512,
                512))

        self.up4 = nn.ConvTranspose2d(512, 256, 2, stride=2)
        self.dec4 = nn.Sequential(CBR(512, 256), CBR(256,
                256))

        self.up3 = nn.ConvTranspose2d(256, 128, 2, stride=2)
        self.dec3 = nn.Sequential(CBR(256, 128), CBR(128,
                128))

        self.up2 = nn.ConvTranspose2d(128, 64, 2, stride=2)
        self.dec2 = nn.Sequential(CBR(128, 64), CBR(64, 64))

        self.up1 = nn.ConvTranspose2d(64, 32, 2, stride=2)
        self.dec1 = nn.Sequential(CBR(64, 32), CBR(32, 32))

        self.final = nn.Conv2d(32, 1, 1)

    def forward(self, x):
        e1 = self.enc1(x)
        e2 = self.enc2(self.pool(e1))
        e3 = self.enc3(self.pool(e2))
        e4 = self.enc4(self.pool(e3))

        m = self.middle(self.pool(e4))
```

```
        d4 = self.up4(m)
        d4 = torch.cat([d4, e4], dim=1)
        d4 = self.dec4(d4)

        d3 = self.up3(d4)
        d3 = torch.cat([d3, e3], dim=1)
        d3 = self.dec3(d3)

        d2 = self.up2(d3)
        d2 = torch.cat([d2, e2], dim=1)
        d2 = self.dec2(d2)

        d1 = self.up1(d2)
        d1 = torch.cat([d1, e1], dim=1)
        d1 = self.dec1(d1)

        out = torch.sigmoid(self.final(d1))
        return out
```

```
1 device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
2 model = UNetMini().to(device)
3
4 optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)
5 criterion = nn.BCELoss()
```

Figura 4.6: Inicialización del modelo Mini U-Net .

Ambos modelos han sido implementados en PyTorch de forma modular y simétrica, y utilizan convoluciones con `padding=1`, `Batch Normalization`, activaciones ReLU y upsampling mediante `ConvTranspose2d`.

Cuadro 4.3: Comparativa detallada entre las arquitecturas U-Net y U-Net Mini.

Característica	**U-Net**	**U-Net Mini**
Canales iniciales	64	32
Capas de codificación	4 bloques (hasta 512 canales)	4 bloques (hasta 256 canales)
Capas intermedias (bottleneck)	2 capas de 1024 canales	2 capas de 512 canales
Capas de decodificación	4 bloques simétricos	4 bloques simétricos
Función de activación	ReLU	ReLU
Normalización	Batch Normalization	Batch Normalization
Padding	Sí (padding=1)	Sí (padding=1)
Upsampling	ConvTranspose2d	ConvTranspose2d
Conexiones skip	Sí (concatenación)	Sí (concatenación)
Capa final	Convolución 1×1 + sigmoid	Convolución 1×1 + sigmoid
Tipo de salida	Más precisa, más pesada	Más ligera, menor precisión
Número total de parámetros	$\sim$31 millones	$\sim$7.7 millones
Tamaño del modelo (params)	118.42 MB	29.62 MB
Tamaño del input	$128 \times 128 \times 3$	$128 \times 128 \times 3$
Tamaño total en memoria (estimado)	320.48 MB	130.81 MB
Velocidad de entrenamiento	Más lento	Más rápido
Uso en dispositivos limitados	Menos eficiente	Más adecuado

El entrenamiento se llevó a cabo durante varias épocas utilizando un esquema de aprendizaje supervisado. En cada iteración, el modelo ajusta sus parámetros internos minimizando una función de pérdida, concretamente la Binary Cross-Entropy, que cuantifica la discrepancia entre las predicciones del modelo y las máscaras reales.

Durante el entrenamiento, se observó una mejora progresiva en la precisión del modelo, reflejada en la disminución de la pérdida y el aumento de las métricas de segmentación. Esta evolución positiva confirma que el modelo ha aprendido a

identificar correctamente las regiones ocupadas por paneles solares en imágenes no vistas durante el entrenamiento.

Con el fin de evitar el sobreajuste y optimizar el proceso de entrenamiento, se ha utilizado la técnica de early stopping. Este mecanismo permite interrumpir automáticamente el entrenamiento cuando el rendimiento sobre el conjunto de validación deja de mejorar tras un número determinado de épocas consecutivas. En este caso, se ha fijado una paciencia de 8 épocas, es decir, si la pérdida de validación no mejora durante 8 épocas seguidas, el entrenamiento se detiene.

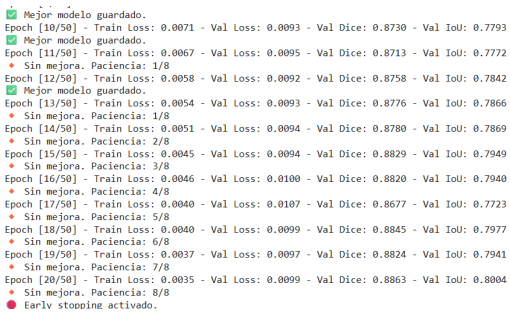

Figura 4.7: Activación de *early stopping* tras 8 épocas sin mejora en la pérdida de validación.

Esta estrategia permite:

- Reducir el tiempo de cómputo.
- Conservar la capacidad de generalización del modelo.
- Prevenir el sobreajuste a ruido o patrones espurios del conjunto de entrenamiento.

Por otro lado, en esta implementación, los hiperparámetros principales (número de canales, tasa de aprendizaje, función de pérdida, optimizador, etc.) se han definido manualmente a partir de experiencia previa y validación empírica. Sin embargo, para estudios más sistemáticos o experimentos escalables, puede integrarse una herramienta de optimización de hiperparámetros como:

- **Keras Tuner** (en entornos basados en TensorFlow/Keras)
- **Optuna**, **Ray Tune** o **Weights & Biases Sweeps** (compatibles con PyTorch)

Estas herramientas permiten explorar de forma automática combinaciones de hiperparámetros mediante métodos de búsqueda aleatoria, optimización bayesiana o algoritmos evolutivos. Su integración permite acelerar el diseño de modelos óptimos y mejorar el rendimiento sin depender exclusivamente de ensayo y error manual.

Los resultados preliminares obtenidos durante el proceso de entrenamiento evidencian un comportamiento adecuado de la arquitectura U-Net en la segmentación de paneles solares, incluso en escenarios visualmente complejos como entornos urbanos heterogéneos. No obstante, el análisis detallado de su rendimiento se desarrollará en los apartados posteriores.

4.1.5. Transfer learning de ResNet34 + U-Net

Con la intención de optimizar los resultados obtenidos y, de forma especialmente relevante, reducir significativamente el tiempo de cómputo durante el entrenamiento, se ha optado por incorporar técnicas de transfer learning en la arquitectura U-Net.

En concreto, se ha integrado como codificador (encoder) una red ResNet34 preentrenada sobre el conjunto de datos ImageNet, ampliamente utilizado en tareas de visión por computador. En este caso, únicamente se ha empleado el tronco convolucional de ResNet34 como extractor de características, omitiendo su capa de clasificación final. Esta estructura actúa como encoder dentro de la U-Net, aportando representaciones visuales enriquecidas que capturan patrones espaciales y texturales relevantes para la tarea.

Cuadro 4.4: Resumen de la arquitectura U-Net + codificador ResNet34

Componente	**Descripción / Salida**	**Parámetros**
Codificador (ResNet34)	4 bloques secuenciales, hasta 512 canales	18.1M
Decodificador (U-Net)	5 bloques de upsampling, hasta 16 canales	3.15M
Capa final	Conv2d 1×1 + activación sigmoid	145
Total de parámetros		**24.44M**
Parámetros entrenables		**24.44M**
No entrenables		0
Tamaño de entrada	$128 \times 128 \times 3$	
Tamaño forward/backward	Memoria requerida para una pasada	35.91 MB
Tamaño del modelo (params)	Memoria ocupada por los pesos	97.75 MB
Tamaño total estimado	Incluyendo activaciones + pesos	133.86 MB

```python
# Modelo U-Net con ResNet34 preentrenado
model = smp.Unet(
    encoder_name="resnet34",        # Encoder preentrenado en ImageNet
    encoder_weights="imagenet",
    in_channels=3,                  # 3 canales RGB
    classes=1,                      # Salida binaria
).to('cuda' if torch.cuda.is_available() else 'cpu')
```

Figura 4.8: Inicialización del modelo U-Net con encoder ResNet34 preentrenado en ImageNet.

A partir de dichas representaciones, el decodificador de U-Net se encarga de reconstruir la segmentación a nivel de píxel, aprovechando las skip connections para recuperar la resolución espacial original perdida en el proceso de codificación.

Para entrenar este modelo, se ha utilizado el optimizador Adam con una tasa de aprendizaje inicial de 1×10^{-4}. Adam (*Adaptive Moment Estimation*) es un método de optimización estocástico ampliamente utilizado en redes neuronales profundas, ya que combina las ventajas de los algoritmos AdaGrad y RMSProp. Calcula adaptativamente tasas de aprendizaje individuales para cada parámetro

del modelo a partir de estimaciones de los momentos de primer y segundo orden (media y varianza) de los gradientes.

Asimismo, se ha utilizado como función de pérdida la *Binary Cross-Entropy with Logits* (`BCEWithLogitsLoss`), la cual integra de forma conjunta la activación sigmoide y la función de entropía cruzada binaria. Esta formulación es preferible desde el punto de vista computacional, ya que mejora la estabilidad numérica del entrenamiento y permite calcular directamente la pérdida a partir de las salidas del modelo, sin necesidad de aplicar la función sigmoide de forma explícita.

A diferencia de la función `BCELoss`, utilizada en la implementación original de U-Net, que requiere aplicar manualmente la activación sigmoide a la salida del modelo, `BCEWithLogitsLoss` evita posibles errores de redondeo y mejora la eficiencia del entrenamiento. Esta elección resulta especialmente adecuada cuando se emplean arquitecturas como `Unet` con codificadores preentrenados (por ejemplo, `ResNet34`), cuyas salidas no están normalizadas y requieren una función de pérdida robusta frente a valores extremos.

```
# Definir optimizador y pérdida
optimizer = optim.Adam(model.parameters(), lr=1e-4)
criterion = nn.BCEWithLogitsLoss()
```

Figura 4.9: Definición del optimizador Adam y la función de pérdida.

De este modo, se espera que esta combinación entre U-Net y ResNet34 preentrenada contribuya a:

- Reducir el tiempo de entrenamiento y la necesidad de grandes volúmenes de datos.
- Mejorar la capacidad de generalización del modelo ante imágenes variadas.
- Aumentar la precisión de la segmentación al partir de características visuales sólidas.

No obstante, la efectividad de este enfoque será analizada en detalle en los apartado siguiente, donde se evaluarán empíricamente los resultados obtenidos.

4.1.6. Análisis de resultados y comparación de modelos

Con el fin de identificar la arquitectura más eficaz para la segmentación automática de paneles solares, se ha llevado a cabo una comparación entre las tres variantes del modelo U-Net desarrolladas: una implementación base, una versión

reducida (U-Net Mini) y una tercera que incorpora transfer learning mediante un encoder ResNet34 preentrenado en ImageNet.

A continuación se resumen los resultados obtenidos para cada modelo tras aplicar la estrategia de early stopping y evaluar las métricas sobre el conjunto de validación:

Cuadro 4.5: Comparativa de rendimiento entre modelos de segmentación

Modelo	**Parámetros**	**Val. Dice**	**Val. IoU**	**Epoch óptima**
U-Net	~31 M	**0.9058**	**0.8319**	29
U-Net Mini	~7 M	0.8863	0.8004	20
U-Net + ResNet34	~23 M	0.8653	0.7709	9

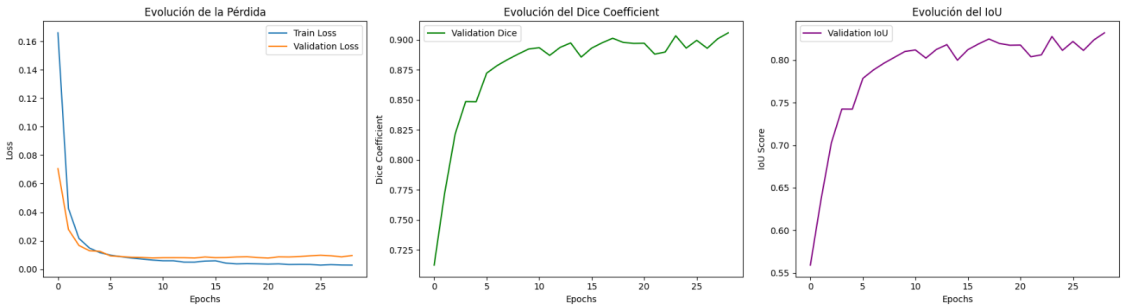

Figura 4.10: Evolución de las métricas de evaluación de Unet

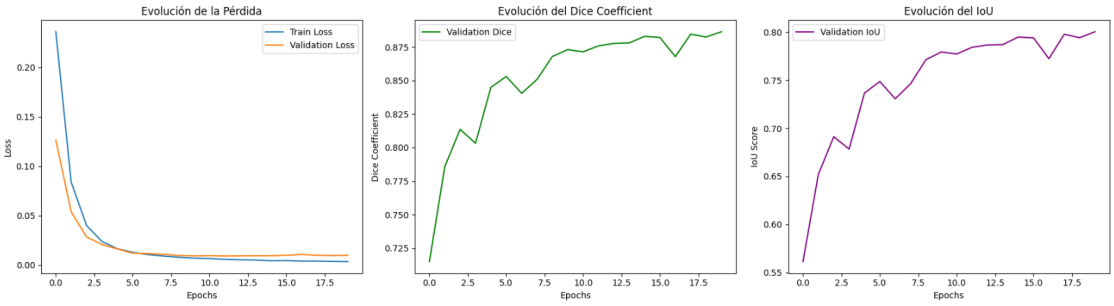

Figura 4.11: Evolución de las métricas de evaluación de MiniUnet

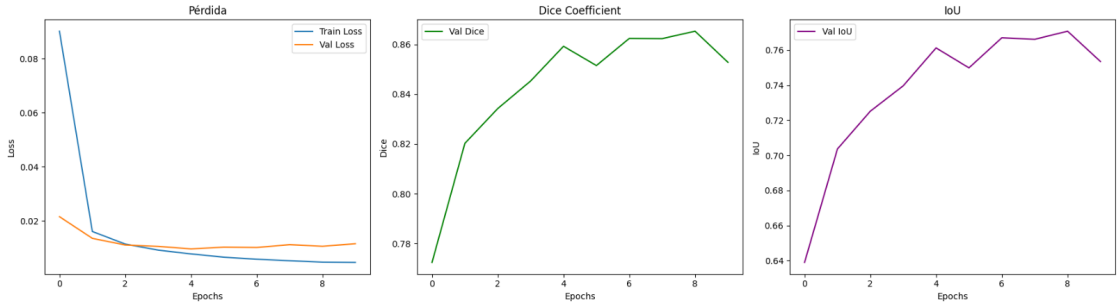

Figura 4.12: Evolución de las métricas de evaluación de Unet + Resnet34

Desde el punto de vista cuantitativo, la U-Net estándar ha sido el modelo que ha alcanzado los mejores valores de Dice (0.9058) y IoU (0.8319), superando tanto a su versión compacta como a la arquitectura basada en transfer learning.

La U-Net Mini, aunque presenta una capacidad de representación reducida, ha demostrado un comportamiento notable, con métricas apenas inferiores y un coste computacional mucho menor. Por su parte, el modelo U-Net con encoder ResNet34 ha logrado converger rápidamente y mostrar estabilidad, aunque sin superar los resultados de la arquitectura original.

A pesar de que se esperaba que la incorporación de técnicas de transfer learning mediante el uso de ResNet34 como codificador mejorase el rendimiento del modelo, los resultados cuantitativos obtenidos no reflejan una mejora sustancial respecto a la arquitectura clásica de U-Net.

Para analizar el comportamiento interno de cada modelo durante la segmentación, se ha utilizado la técnica Grad-CAM (Gradient-weighted Class Activation Mapping). Este método permite visualizar qué regiones de la imagen han tenido mayor influencia en la predicción, generando un mapa de calor a partir de los gradientes de activación.

Grad-CAM calcula un mapa de atención sobre una capa convolucional específica (habitualmente del encoder), combinando sus mapas de activación A^k con los gradientes $\frac{\partial y^c}{\partial A^k}$ respecto a la clase objetivo c.

El peso de cada canal se define como:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{4.3}$$

y el mapa Grad-CAM resultante es:

$$L^c_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha^c_k A^k\right) \tag{4.4}$$

Donde Z es el número de elementos espaciales, y la función ReLU elimina las contribuciones negativas.

La siguiente función en PyTorch genera visualizaciones Grad-CAM sobre cuatro imágenes de entrada, aplicando hooks manuales para extraer activaciones y gradientes en una capa del encoder:

Listing 4.3: Función para visualizar Grad-CAM sobre U-Net

```
def visualize_gradcam_batch_unet(model, image_batch, layer='enc4',
        device='cuda'):
    model.eval()
    image_batch = image_batch.to(device)[:4]
    activations, gradients = {}, {}

    def forward_hook(module, input, output):
        activations['value'] = output.detach()

    def backward_hook(module, grad_input, grad_output):
        gradients['value'] = grad_output[0].detach()

    target_layer = getattr(model, layer)
    h1 = target_layer.register_forward_hook(forward_hook)
    h2 = target_layer.register_full_backward_hook(backward_hook)

    outputs = model(image_batch)
    preds = torch.sigmoid(outputs).detach().cpu().squeeze(1).numpy
        ()

    for i in range(4):
        model.zero_grad()
        output = outputs[i:i+1]
        output.mean().backward(retain_graph=True)

        grads = gradients['value'][i]
        acts = activations['value'][i]
        weights = grads.mean(dim=(1, 2), keepdim=True)
        cam = torch.relu((weights * acts).sum(dim=0))
        cam /= cam.max() + 1e-8
        cam_np = cam.cpu().numpy()

        img = image_batch[i].cpu().permute(1, 2, 0).numpy()
        cam_resized = cv2.resize(cam_np, (img.shape[1], img.shape
            [0]))
```

```
        # Visualizacion omitida
        ...
    h1.remove()
    h2.remove()
```

Esta función permite interpretar visualmente si la atención del modelo está correctamente enfocada en las regiones donde se localizan los paneles solares. Las visualizaciones revelan diferencias notables en la atención espacial:

- La arquitectura **U-Net original** muestra activaciones bien localizadas y centradas en las estructuras de los tejados donde se encuentran los paneles solares, lo que evidencia una capacidad de detección precisa y consistente.

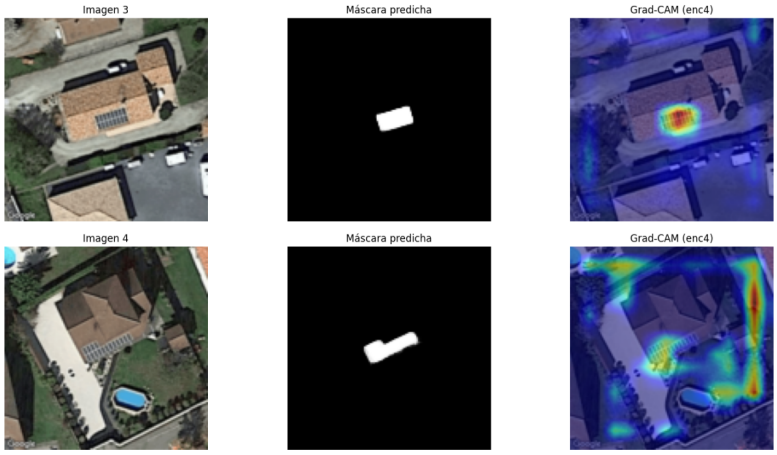

Figura 4.13: Predicción y mapa de activación del modelo **UNet** sobre dos ejemplos.

- La **U-Net Mini**, aunque presenta una segmentación eficaz, tiende a focalizarse en regiones más amplias, incluyendo zonas circundantes no relevantes, lo que puede explicar su ligera pérdida de precisión.

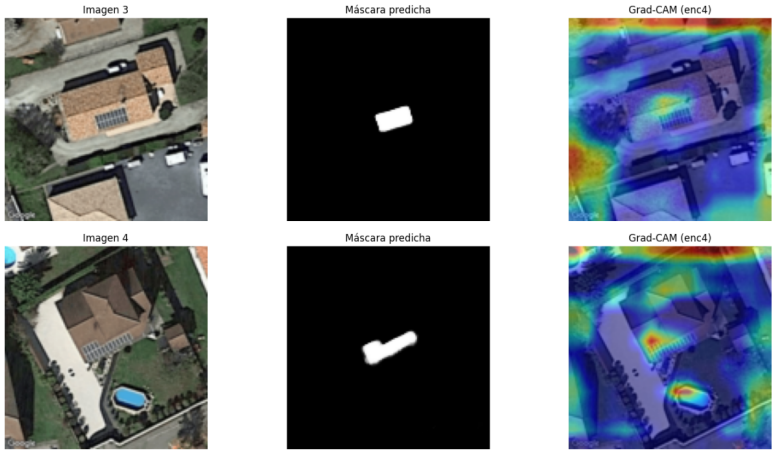

Figura 4.14: Predicción y mapa de activación del modelo **MiniUNet** sobre dos ejemplos.

- En el caso de **U-Net con ResNet34**, las activaciones aparecen más dispersas y, en algunos casos, ambiguas. Esta atención menos específica podría deberse a que el encoder preentrenado en ImageNet extrae características visuales más generales, que no siempre se ajustan óptimamente a tejados residenciales.

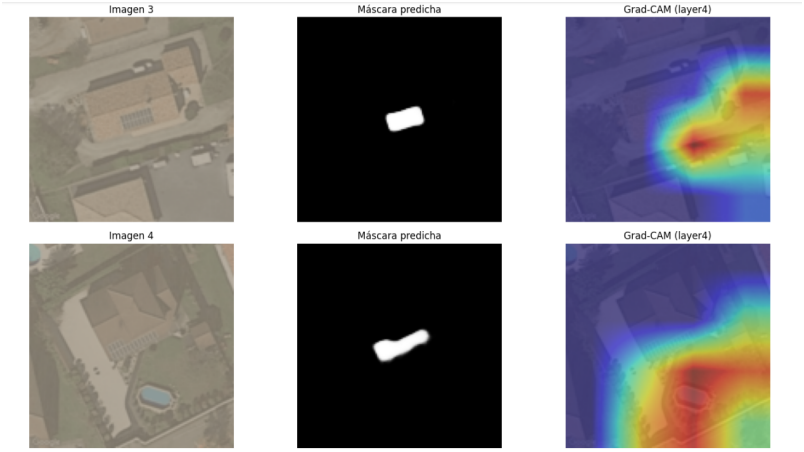

Figura 4.15: Predicción y mapa de activación del modelo **Resnet34 + UNet** sobre dos ejemplos.

Estas observaciones sugieren que, si bien el transfer learning puede ofrecer ventajas iniciales en contextos con escasez de datos, su eficacia no está garantizada cuando se dispone de un conjunto de entrenamiento amplio y representativo. En

este caso concreto, la arquitectura U-Net estándar se posiciona como la opción más equilibrada en términos de precisión, estabilidad y capacidad de generalización.

A continuación se resumen las caracterísiticas mas relevantes de los modelos entreandos:

Cuadro 4.6: Comparativa de arquitecturas -Net (configuración y entrenamiento)

Modelo	**¿Preentrenado?**	**Entrenamiento**
U-Net estándar	No	Desde cero (completo)
Mini U-Net	No	Desde cero (completo)
U-Net + ResNet34	Sí (ImageNet)	Transfer learning parcial (encoder congelado o parcialmente ajustado)

Cuadro 4.7: Comparativa de arquitecturas U-Net (ventajas y limitaciones)

Modelo	**Ventajas observadas**	**Limitaciones observadas**
U-Net estándar	Alta precisión (Dice 0.9058), detección clara de bordes, segmentación definida	Costoso computacionalmente, entrenamiento más lento
Mini U-Net	Eficiente y rápido, menos de $\frac{1}{4}$ de parámetros, ideal para despliegue ligero	Menor precisión en bordes y detalles, mayor ruido en predicción
U-Net + ResNet34	Convergencia rápida, buena estabilidad inicial	Segmentación más difusa, atención menos específica, sin mejora real con respecto a U-Net clásica. Aquí el transfer learning no fue eficaz porque el dominio (tejados con paneles) es muy distinto al de ImageNet

5. EVALUACIÓN DEL MODELO

Con el objetivo de validar la eficacia del modelo desarrollado en un entorno realista, se ha adoptado una estrategia basada en la combinación de dos fuentes de datos. Por un lado, el entrenamiento del modelo se ha realizado utilizando un dataset público de imágenes aéreas y máscaras de segmentación procedente de distintas regiones de Francia [14], que permite aplicar técnicas supervisadas con una base sólida y controlada.

Por otro lado, para evaluar el rendimiento del sistema en un contexto geográfico y visual distinto, se han utilizado imágenes aéreas reales del municipio español de Molina de Segura, obtenidas mediante Google Earth Pro [24]. Esta combinación permite medir la capacidad de generalización del modelo y su aplicabilidad práctica en escenarios reales.

En concreto, para la fase de validación en entorno real, se han empleado 25 imágenes con sus correspondientes máscaras de segmentación manual, lo que representa aproximadamente un **0,12 %** del total de imágenes utilizadas durante el entrenamiento del modelo. Este número limitado de datos de prueba se debe al elevado coste temporal y manual que implica tanto la obtención de imágenes georreferenciadas como la generación precisa de sus máscaras asociadas.

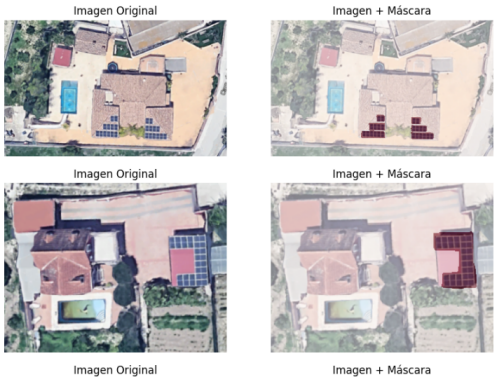

Figura 5.1: Ejemplo de imágenes obtenidas con su mascara asociada creada.

Las imágenes fueron recopiladas a través de la plataforma Google Earth Pro, mientras que las máscaras correspondientes fueron generadas manualmente utilizando la herramienta Label Studio, que permite una anotación detallada y precisa a nivel de píxel. Este conjunto reducido, pero cuidadosamente preparado, permite una evaluación cualitativa y cuantitativa robusta del modelo en un caso de aplicación real.

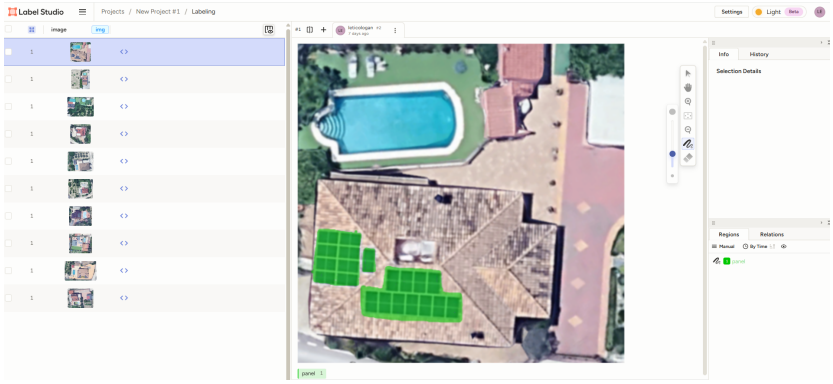

Figura 5.2: Interfaz Label Studio para la creación de las máscaras asociadas.

5.1. Justificación de la elección del área de estudio

La selección del municipio de Molina de Segura como área piloto para la detección automática de instalaciones fotovoltaicas en viviendas unifamiliares responde a una combinación de factores técnicos, climáticos, urbanísticos y de disponibilidad de datos que lo convierten en un caso representativo y viable.

Figura 5.3: Localización de Molina de Segura.

- **Radiación solar y potencial energético:** La Región de Murcia, donde se sitúa el municipio, registra más de 3.000 horas de sol al año [25], lo que la

convierte en una zona óptima para el autoconsumo solar desde el punto de vista técnico y económico.

- **Tipología urbanística favorable:** Molina de Segura cuenta con una elevada proporción de viviendas unifamiliares y adosadas, con tejados inclinados y orientaciones adecuadas, características favorables tanto para la detección automática como para la instalación efectiva de paneles.
- **Calidad de los datos disponibles:** La disponibilidad de imágenes aéreas frecuentes y actualizadas mediante herramientas como Google Earth Pro [24] facilita la aplicación de modelos de segmentación, permitiendo detectar paneles y analizar variables geométricas relevantes.
- **Apoyo institucional y contexto normativo:** El municipio impulsa políticas activas de sostenibilidad, incluyendo bonificaciones sobre el IBI para viviendas con sistemas solares [26], lo que refuerza el interés aplicado del proyecto.
- **Escalabilidad y representatividad:** Con unos 70.000 habitantes, Molina ofrece un entorno urbano manejable y extrapolable a otros municipios del sureste español, combinando complejidad técnica y viabilidad logística.

Por todo lo anterior, Molina de Segura constituye un escenario de estudio idóneo, con alto potencial de impacto medioambiental, relevancia científica y aplicabilidad real en la planificación energética y estrategias de transición ecológica basadas en datos.

5.2. Obtención de resultados

En este apartado se evalúa la capacidad de generalización de los modelos previamente entrenados al ser aplicados sobre el nuevo conjunto de imágenes reales del municipio de Molina de Segura.

El primer modelo analizado ha sido la arquitectura U-Net entrenada desde cero con el conjunto de datos base. Al aplicarla sobre las imágenes de Molina de Segura, los resultados obtenidos mostraron un **coeficiente Dice promedio de 0.0278** y una **IoU media de 0.0146**. Estas métricas reflejan un rendimiento limitado, con escasa superposición entre las predicciones del modelo y las máscaras reales.

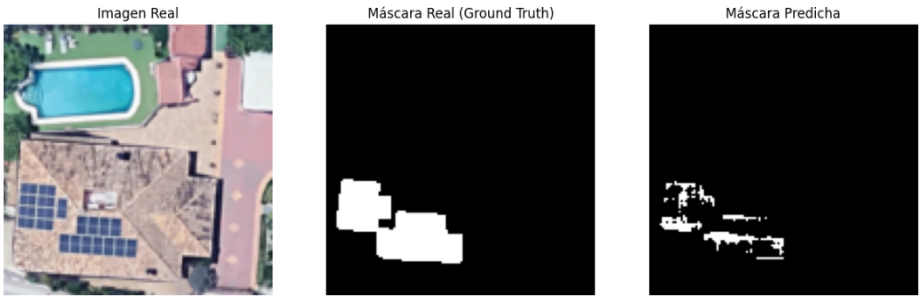

Figura 5.4: Predicción de U-Net sobre una imagen de Molina de Segura (imagen, máscara predicha)

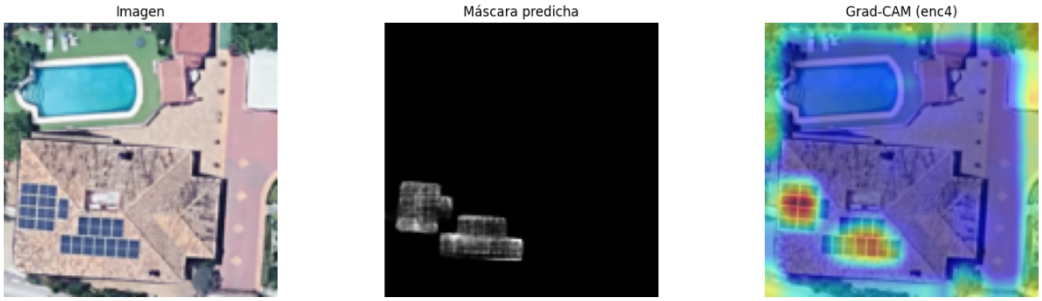

Figura 5.5: Predicción de U-Net sobre una imagen de Molina de Segura (imagen, máscara predicha y mapa Grad-CAM)

La Figura 5.5 muestra un ejemplo representativo del comportamiento del modelo. A pesar de que se observan ciertas activaciones en zonas relevantes del tejado, la segmentación final presenta numerosos errores de forma y fragmentación, sin lograr una detección precisa y coherente de las placas solares. El mapa de activación generado con Grad-CAM indica que el modelo sí logra centrar su atención en áreas de interés, pero la información extraída no resulta suficientemente discriminativa en este nuevo contexto.

Este comportamiento pone de manifiesto los desafíos que implica la generalización en tareas de segmentación semántica, especialmente cuando existen variaciones en iluminación, resolución, materiales o tipologías arquitectónicas.

A continuación, se evaluó el rendimiento de la arquitectura Mini U-Net sobre el conjunto de validación compuesto por 25 imágenes reales del municipio de Molina de Segura. Este modelo se caracteriza por una estructura más ligera y eficiente, lo que reduce considerablemente el tiempo de entrenamiento y los requisitos computacionales.

Los resultados obtenidos reflejan una capacidad limitada de generalización en el nuevo entorno geográfico. Las métricas medias alcanzadas fueron un coeficiente de Dice de 0.0278 y una puntuación IoU de 0.0146, valores que, al igual que en el modelo U-Net original, evidencian las dificultades del modelo para adaptarse a las particularidades visuales del nuevo dominio, como el estilo de los tejados, la iluminación o la resolución espacial.

En la Figura 5.6, se muestra un ejemplo de predicción donde, si bien se detecta parcialmente la ubicación de los paneles solares, la máscara generada presenta ruido y una segmentación poco precisa. No obstante, el mapa de activación Grad-CAM revela que el modelo focaliza su atención en la región correcta del tejado, lo cual sugiere que la extracción de características es adecuada, pero el decodificador no logra refinar la segmentación con suficiente precisión.

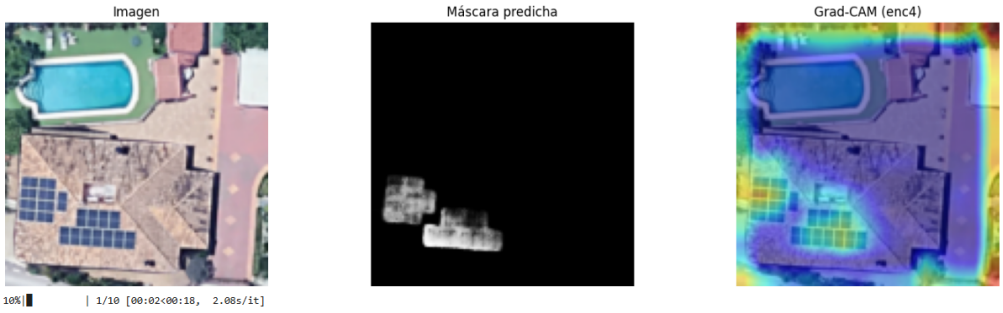

Figura 5.6: Predicción de Mini U-Net sobre una imagen de Molina de Segura (imagen, máscara predicha y mapa Grad-CAM).

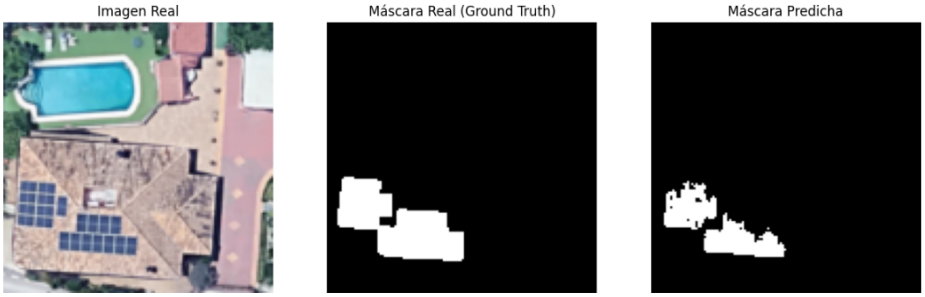

Figura 5.7: Predicción de Mini U-Net sobre una imagen de Molina de Segura (imagen, máscara predicha).

Cabe destacar que las métricas obtenidas pueden estar condicionadas por el reducido tamaño del conjunto de validación y su falta de representatividad estadística. Además, el cambio de dominio geográfico y visual respecto al conjunto

de entrenamiento puede haber afectado negativamente al rendimiento del modelo. A esto se suma la creación manual de las máscaras de validación lo que puede introducir cierto grado de imprecisión y afectar la evaluación cuantitativa.

Finalmente, se evaluó el modelo U-Net con codificador ResNet34 preentrenado, entrenado bajo un esquema de transfer learning. Este modelo alcanzó unas métricas notablemente superiores al aplicar la validación sobre las imágenes de Molina de Segura, obteniendo un **coeficiente Dice promedio de 0.6620** y una **IoU media de 0.6444**. A primera vista, estos valores sugerirían un rendimiento excelente.

Sin embargo, el análisis cualitativo de las segmentaciones obtenidas revela una notable discrepancia con respecto a las métricas. A pesar de los valores elevados de Dice e IoU, las máscaras predichas presentan formas borrosas, escasa definición y, en ocasiones, detecciones incorrectas o irrelevantes. Esto sugiere que el modelo podría estar sobreajustado a patrones no significativos o bien estar captando estructuras genéricas que coinciden parcialmente con las máscaras reales, sin representar verdaderamente los paneles solares.

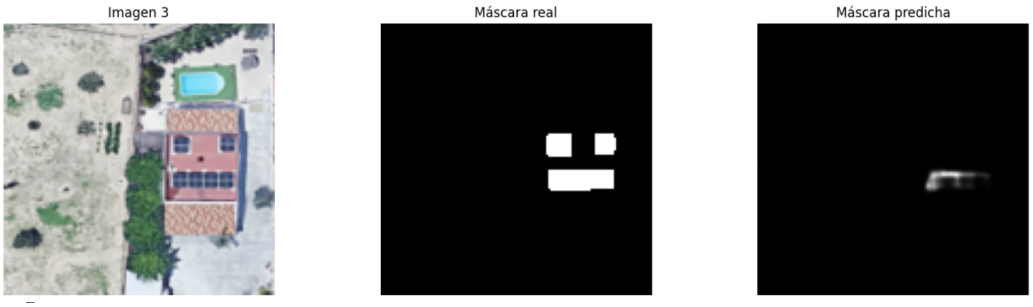

Figura 5.8: Predicción de U-Net +ResNet34 sobre una imagen de Molina de Segura (imagen, máscara predicha).

Figura 5.9: Predicción de U-Net +ResNet34 sobre una imagen de Molina de Segura (imagen, máscara predicha y mapa Grad-CAM).

Este fenómeno podría explicarse por varios factores:

- Las buenas métricas pueden deberse a ciertas coincidencias globales sin que haya una segmentación precisa.
- La función de pérdida utilizada (`BCEWithLogitsLoss`) puede favorecer predicciones suaves o extendidas si la máscara real es relativamente uniforme.
- El encoder ResNet34, al estar preentrenado en un dominio genérico (ImageNet), podría no haber sido suficientemente adaptado al nuevo dominio de tejados fotovoltaicos.

En conjunto, estos resultados ponen de relieve la necesidad de combinar evaluaciones cuantitativas y cualitativas en tareas de segmentación semántica, especialmente cuando se evalúan modelos bajo un cambio de dominio. A pesar de las métricas aparentemente satisfactorias, una inspección visual revela que la segmentación obtenida dista de ser funcional en un contexto práctico.

Este contraste entre métricas elevadas y resultados visuales deficientes subraya una problemática habitual en la evaluación de modelos de segmentación semántica: la posibilidad de que los indicadores cuantitativos no reflejen fielmente la calidad perceptiva o la utilidad real de las predicciones. En contextos aplicados como la detección de paneles solares, donde la precisión espacial y la interpretabilidad son fundamentales, la validez práctica de un modelo no puede limitarse a cifras globales.

Por tanto, se evidencia la importancia de incorporar inspecciones visuales sistemáticas y análisis más finos en las fases de validación, especialmente cuando

se trabaja con dominios geográficos distintos al del entrenamiento. En este caso concreto, los resultados sugieren que, aunque el uso de *transfer learning* con ResNet34 mejora la capacidad del modelo para identificar patrones generales, sigue siendo necesaria una adaptación más profunda al dominio específico de tejados fotovoltaicos del sureste español.

Cuadro 5.1: Resumen de evaluación de los modelos en Molina de Segura

Modelo	**Dice / IoU**	**Comportamiento cuantitativo**	**Observaciones cualitativas**	**Limitaciones detectadas**
U-Net estándar	0.0278 / 0.0146	Métricas muy bajas; casi nula superposición con las máscaras reales	Segmentación pobre; activaciones centradas pero sin buena discriminación	Generalización fallida; el modelo no se adapta a nuevas condiciones visuales
Mini U-Net	0.0278 / 0.0146	Mismas métricas que la U-Net; sin mejora pese a su ligereza	Activación correcta pero segmentación imprecisa; ruido elevado	Falta de capacidad para refinar la salida; baja resolución en detalles
U-Net + ResNet34	**0.6620 / 0.6444**	Métricas altas a primera vista; mejor que los otros modelos	Segmentación borrosa; formas poco definidas y detecciones irrelevantes	Posible sobreajuste; el modelo aprende patrones no útiles

En el capítulo siguiente se abordarán estas limitaciones en mayor detalle, así como posibles estrategias para mejorar la robustez y aplicabilidad de los modelos en entornos reales

6. CONCLUSIONES

Tras el desarrollo y validación del sistema propuesto para la detección automática de instalaciones solares mediante segmentación semántica en imágenes aéreas, se han obtenido resultados relevantes que permiten reflexionar sobre los logros alcanzados y las barreras encontradas. A continuación, se exponen las principales limitaciones identificadas durante el proceso, así como una serie de propuestas para su abordaje en trabajos futuros.

6.1. Limitaciones

A lo largo del desarrollo de este proyecto, se han identificado diversas limitaciones que han condicionado tanto el diseño del sistema como su implementación práctica. Estas limitaciones, si bien no han impedido el cumplimiento de los objetivos propuestos, sí han supuesto retos relevantes que conviene destacar.

En primer lugar, una de las principales dificultades fue la identificación de un dataset público adecuado para el entrenamiento del modelo. La mayoría de conjuntos de datos disponibles relacionados con energía solar presentan restricciones de acceso, están orientados a contextos geográficos muy específicos o no incluyen máscaras de segmentación suficientemente detalladas. Tras un proceso exhaustivo de búsqueda y evaluación, se seleccionó finalmente un dataset de imágenes aéreas procedentes de distintas regiones de Francia [14], que ofrecía un equilibrio adecuado entre calidad, cobertura geográfica y disponibilidad de anotaciones.

En segundo lugar, la capacidad de cómputo local del alumno resultó ser una limitación importante a la hora de entrenar redes neuronales convolucionales de cierta complejidad, como U-Net o variantes con backbones preentrenados.

Debido a la alta demanda computacional asociada al procesamiento de imágenes de alta resolución, se optó por suscribirse a la plataforma Google Colab Pro, lo cual permitió acceder a recursos de GPU avanzados. Esta decisión mejoró significativamente los tiempos de entrenamiento, aunque estos seguían siendo elevados

debido a la naturaleza intensiva de las tareas de segmentación semántica.

Por último, cabe destacar la dificultad asociada a la obtención de un conjunto de datos de validación en un entorno real. La recopilación de imágenes aéreas actualizadas y con suficiente calidad ya representa por sí misma una tarea laboriosa, a lo que se suma el coste humano y temporal de generar manualmente las máscaras de segmentación asociadas. Esta última tarea, que requiere precisión a nivel de píxel, fue abordada mediante la herramienta Label Studio, pero se vio limitada en escala debido a los recursos disponibles. Como consecuencia, el conjunto de validación ha sido de tamaño reducido, aunque suficiente para realizar una evaluación cualitativa representativa.

Estas limitaciones ponen de manifiesto la importancia de contar con recursos adecuados ,tanto en términos de datos como de infraestructura computacional, para el desarrollo y validación de sistemas basados en inteligencia artificial, especialmente en contextos aplicados como la segmentación de instalaciones solares en entornos urbanos.

6.2. Líneas futuras

Con base en las dificultades enfrentadas, se identifican varias estrategias que podrían implementarse en futuros desarrollos para mitigar las limitaciones detectadas:

- **Ampliación del acceso a datos**: Colaborar con instituciones públicas o privadas (como ayuntamientos, empresas del sector energético o universidades) podría facilitar el acceso a datasets más representativos, actualizados y con anotaciones de mayor calidad.
- **Cruce con bases de datos externas para aplicaciones comerciales**: Una posible evolución del sistema sería su integración con bases de datos externas con el fin de identificar y contactar potenciales clientes interesados en la instalación de sistemas fotovoltaicos. Esta funcionalidad permitiría una aplicación directa del modelo en campañas comerciales o estrategias de transición energética. No obstante, esta línea de trabajo debería abordarse con especial atención a los aspectos legales y éticos asociados a la privacidad y protección de datos personales.
- **Automatización parcial de la anotación**: Herramientas de segmentación semiautomática basadas en modelos preentrenados o técnicas de anotación

asistida podrían reducir significativamente el esfuerzo requerido para generar máscaras precisas.

- **Uso de plataformas con mayor capacidad computacional**: Alternativas como Amazon SageMaker, Google Cloud AI Platform o la integración con servidores universitarios con GPU podrían permitir entrenamientos más complejos, con arquitecturas más profundas o resolución de imagen completa.

Estas mejoras permitirían aumentar tanto la escalabilidad como la precisión del sistema desarrollado, allanando el camino para aplicaciones reales a mayor escala.

Además de las mejoras operativas mencionadas, el proyecto abre la puerta a múltiples líneas futuras que podrían complementar y ampliar sus capacidades:

- **Estimación del potencial solar por vivienda**: Integrar el modelo de detección con datos climáticos, de orientación y superficie útil permitiría calcular automáticamente el potencial de generación energética de cada tejado.
- **Clasificación del tipo de instalación**: A partir de las máscaras segmentadas, podría entrenarse un segundo modelo para distinguir entre instalaciones aisladas, agrupadas o industriales, mejorando la capacidad de análisis.
- **Detección multitemporal**: Usar imágenes de diferentes años o estaciones podría permitir identificar cuándo se han instalado paneles, monitorizar el crecimiento de la energía solar en un área determinada o detectar posibles desmantelamientos.
- **Despliegue como servicio web**: Una evolución natural sería transformar el prototipo en una herramienta accesible desde un portal web, permitiendo a administraciones o empresas energéticas analizar regiones específicas bajo demanda.

Estas líneas de trabajo podrían ampliar significativamente el impacto del sistema propuesto, transformándolo en una herramienta integral para la planificación energética basada en datos.

En resumen, este proyecto demuestra la viabilidad de aplicar técnicas de segmentación semántica para la detección automática de instalaciones solares en entornos urbanos, al tiempo que pone de relieve los desafíos inherentes al cambio de dominio, la disponibilidad de datos y la capacidad de generalización de los modelos.

Si bien los resultados iniciales son prometedores, se hace evidente la necesidad de seguir investigando en estrategias de mejora, como la adaptación de dominios, el enriquecimiento de datasets o la integración de modelos más avanzados. El enfoque desarrollado sienta una base sólida para futuras aplicaciones reales orientadas a la transición energética y la sostenibilidad urbana.

7. PLANIFICACIÓN TEMPORAL

Por úlitmo, esta sección tiene como objetivo exponer de manera estructurada la planificación seguida para la elaboración del proyecto. Para ello, se ha diseñado un cronograma detallado que abarca desde las fases iniciales de investigación hasta la preparación final del informe y la presentación.

La distribución temporal de las tareas ha sido clave para garantizar una evolución coherente del trabajo, permitiendo abordar de forma ordenada tanto los aspectos técnicos como los documentales.

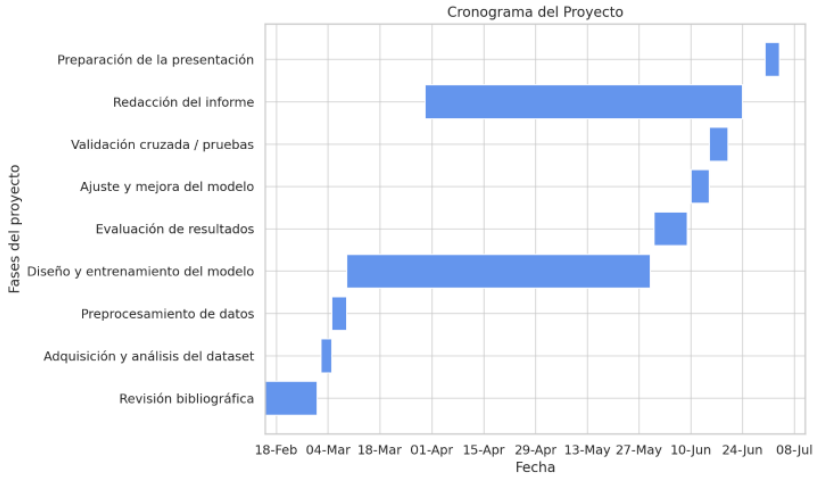

Figura 7.1: Diagrama de Gantt del proyecto.

En la Figura 7.1 se muestra el diagrama de Gantt que resume la secuencia y duración estimada de cada una de las etapas.

BIBLIOGRAFÍA

[1] CanalTIC. Energía solar térmica: cómo funciona y para qué se utiliza, 2024.

[2] Wikipedia contributors. Neurona - wikipedia, la enciclopedia libre, 2023.

[3] Atria Innovation. Esquema de red neuronal artificial, 2023.

[4] Medium Contributors. Esquema del funcionamiento de una red neuronal convolucional (cnn). https://miro.medium.com/v2/resize:fit:1358/1*vkQ0hXDaQv57sALXAJquxA.jpeg, 2023.

[5] Reddit contributors. The u-net neural network architecture for semantic segmentation. https://www.reddit.com/r/learnmachinelearning/comments/voce98/the_unet_neural_network_architecture_for_semantic/, 2022.

[6] Medium Contributors. Esquema visual de transfer learning. https://miro.medium.com/v2/resize:fit:720/format:webp/1*7Ip2_SeOz_BoruHEytEMlQ.png, 2020.

[7] Medium Contributors. Esquema visual de la arquitectura resnet. https://miro.medium.com/v2/resize:fit:1100/format:webp/1*BnoNVpj7uCNMOFOj1DQBQA.png, 2023.

[8] Roboflow Blog. Introduction to imagenet, 2022.

[9] Analytics Vidhya. Visualización del algoritmo k-means clustering. https://cdn.analyticsvidhya.com/wp-content/uploads/2020/10/56854k-means-clustering.webp, 2020.

[10] Ng Hi Huynh. Understanding evaluation metrics in medical image segmentation. https://medium.com/@nghihuynh_37300/understanding-evaluation-metrics-in-medical-image-segmentation-d289a373a3f, 2023.

[11] Jordan M Malof, Leslie M Collins, Kyle Bradbury, and Charles Newell. Automatic detection of solar photovoltaic arrays in high resolution imagery: A comparison of deep learning and random forest approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(5):1930–1943, 2016.

[12] Zihan Yu, F Brabec, K Grolinger, and MA M Capretz. Deep learning-based solar panel detection in aerial imagery. *2018 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*, pages 332–337, 2018.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. pages 234–241, 2015.

[14] Gabriel Kasmi, Yves-Marie Saint-Drenan, David Trebosc, Raphaël Jolivet, Jonathan Leloux, Babacar Sarr, and Laurent Dubus. A crowdsourced dataset of aerial images with annotated solar photovoltaic arrays and installation metadata. *Scientific Data*, 10(1):59, 2023.

[15] Iberdrola. Energía solar: qué es, cómo funciona y por qué es una energía limpia, 2024.

[16] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition, 2021.

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[18] Cyril Voyant, Gilles Notton, Soteris Kalogirou, et al. Machine learning methods for solar radiation forecasting: A review. *Renewable Energy*, 105:569–582, 2017.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[20] Pranav Jain. Understanding u-net: A convolutional network for biomedical image segmentation. `https://towardsdatascience.com/understanding-u-net-61276b10f360`, 2020.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[22] Shruti Jadon. A survey of loss functions for semantic segmentation. In *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–7. IEEE, 2020.

[23] Shruti Jadon. Semsegloss: A python package of loss functions for semantic segmentation. *Software Impacts*, 9:100078, 2021.

[24] Google earth pro. https://www.google.com/earth/versions/, 2025.

[25] IDAE. Atlas de radiación solar en españa. región de murcia. https://murciaencifras.es/datos-curiosos?ref=16&q=Murcia-recibe-casi-3.300-horas-de-sol-al-aNo#:~:text=La%20Regi%C3%B3n%20de%20Murcia%20se,horas%20de%20sol%20al%20a%C3%B1o., 2020.

[26] Trisolar. Bonificaciones para energía solar en molina de segura. https://www.trisolar.es/blog/noticias/bonificaciones-para-energia-solar-en-molina-de-segura/, 2023.