Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Modelo de Deep Learning para el reconocimiento de filas de Viñedos y estimación de parámetros agronómicos desde imágenes satelitales

---

Alumno: Marcos Moreda Blanco

Tutores: José Vicente Álvarez Bravo

Francisco Hernando Gallego

Fecha: 14 de Julio de 2025

Modelo de Deep Learning para el reconocimiento de filas de Viñedos y estimación de parámetros agronómicos desde imágenes satelitales

Marcos Moreda Blanco

Índice general

Índice de figuras

Índice de tablas

Este trabajo va para todas las personas que me han acompañado y apoyado durante estos cinco años. Una parte fundamental de que haya podido llevarlo a cabo es, sin duda, gracias a ellas. Con este trabajo, espero poder devolver, al menos en parte, todo lo que me han dado y demostrar que su ayuda y compañía realmente marcaron la diferencia.

Agradecimientos

Gracias a todos los profesores que he tenido a lo largo de estos años. De cada uno me llevo algo más allá de lo aprendido en sus materias: una lección, una perspectiva o una inspiración. Y en especial a José Vicente y Francisco, los tutores de este TFG, gracias por hacerlo posible. A mis compañeros, gracias por estar siempre dispuestos a ayudar cuando lo he necesitado; compartir este camino con vosotros lo ha hecho mucho más llevadero y enriquecedor.

También quiero agradecer a todos los profesionales de la universidad que, con su trabajo diario, hacen posible su funcionamiento. Gracias a ellos, la educación que he recibido ha sido de gran calidad y el día a día, mucho más fácil.

Y, por supuesto, gracias a mi familia y amigos. Sin vuestro apoyo constante, esta etapa no habría sido posible. Vuestra motivación, comprensión y cariño han sido fundamentales durante todo el proceso.

Un agradecimiento muy especial para mi madre, mi padre y mi hermana. Vuestra ayuda en todos los aspectos de mi vida es algo que nunca podré devolver. Sois el ejemplo perfecto de esfuerzo, generosidad y bondad, valores que intento seguir y por los que me siento increíblemente afortunado de teneros a mi lado.

Resumen

El desarrollo tecnológico y el acceso creciente a datos satelitales han abierto nuevas oportunidades para optimizar prácticas agrícolas a través de sistemas inteligentes. En particular, la viticultura, un sector agrícola de gran relevancia económica en numerosos países, se beneficia cada vez más de herramientas basadas en análisis automatizado de imágenes.

Este trabajo se enmarca en este contexto y tiene como objetivo el diseño y desarrollo de un sistema de inteligencia artificial capaz de identificar, contar y analizar las filas de viñedos a partir de imágenes satelitales, con el fin de extraer métricas agronómicas relevantes como la longitud de las filas, la densidad de plantación y otros parámetros asociados.

Palabras claves: inteligencia artificial, viñedos, U-net, modelo, agricultura, *deep learning*, segmentación, imágenes.

Abstract

Technological advancements and the growing availability of satellite data have opened new opportunities to optimize agricultural practices through intelligent systems. Viticulture, in particular, a key agricultural sector with significant economic importance in many countries, is increasingly benefiting from tools based on automated image analysis.

This work is situated within that context and aims to design and develop an artificial intelligence system capable of identifying, counting, and analyzing vineyard rows from satellite images, in order to extract relevant agronomic metrics such as row length, planting density, and other associated parameters.

Keywords: artificial intelligence, vineyards, U-Net, model, agriculture, deep learning, segmentation, images.

Parte I

Descripción del proyecto

Capítulo 1

Introducción

El desarrollo tecnológico y el acceso creciente a datos satelitales han abierto nuevas oportunidades para optimizar prácticas agrícolas mediante sistemas inteligentes. En particular, la viticultura (un sector agrícola de gran relevancia económica en numerosos países, como España, Francia o Italia) se beneficia cada vez más de herramientas basadas en el análisis automatizado de imágenes.

Este trabajo de fin de grado se enmarca en este contexto y tiene como objetivo el diseño y desarrollo de un sistema de inteligencia artificial capaz de identificar, contar y analizar las filas de viñedos a partir de imágenes satelitales, con el fin de extraer métricas agronómicas relevantes como la longitud de las filas, la densidad de plantación y otros parámetros estructurales. La automatización de este tipo de tareas busca mejorar la eficiencia en la gestión del viñedo y facilitar la toma de decisiones agronómicas con una base más precisa y escalable.

A pesar de que existen herramientas comerciales para el análisis de cultivos, muchas están centradas en el seguimiento de parámetros fisiológicos (como el índice NDVI), pero no abordan con suficiente precisión la estructura del viñedo. En este sentido, el enfoque basado en segmentación semántica mediante redes neuronales convolucionales (como *U-Net*) constituye una línea de investigación prometedora, capaz de superar las limitaciones de los métodos tradicionales y aportar mayor granularidad en la información espacial.

Desde una perspectiva práctica, esta tecnología puede tener un impacto directo en varios frentes. Por ejemplo, permite calcular automáticamente el número de cepas por parcela a partir de la longitud total de las filas, lo que facilita la estimación de la productividad potencial de una explotación. Esto puede traducirse en un ahorro económico al evitar visitas de campo para realizar conteos manuales. Asimismo, una segmentación precisa puede ayudar a detectar zonas improductivas o afectadas por enfermedades, permitiendo intervenciones más focalizadas que reducen el uso innecesario de agua, fertilizantes o productos fitosanitarios. Incluso en procesos posteriores, como la estimación del volumen de vino que se puede obtener en función del número de cepas, este tipo de análisis estructural puede integrarse como herramienta predictiva en la cadena de producción.

En términos medioambientales, el conocimiento preciso de la distribución de las filas también permite diseñar planes de riego más eficientes y sostenibles, evitando el despilfarro

de agua y reduciendo el impacto ecológico de la actividad agrícola.

Este primer capítulo presenta el proyecto en profundidad desde una perspectiva conceptual, organizativa y técnica. Se incluye el planteamiento del problema, los objetivos perseguidos, los condicionantes identificados durante el diseño del sistema, el alcance del trabajo y, finalmente, la estructura que seguirá el resto de la memoria.

1.1. Planteamiento del problema

La agricultura de precisión representa una evolución en la forma en que los agricultores monitorizan y gestionan sus cultivos, permitiendo una toma de decisiones más fundamentada gracias al uso de tecnologías como la teledetección, la inteligencia artificial y el análisis de datos. En el caso particular de los viñedos, la disposición espacial de las plantas (organizadas en filas lineales) ofrece una oportunidad única para automatizar su identificación mediante técnicas de visión por computador.

Actualmente, muchas tareas relacionadas con la planificación, el seguimiento y el análisis de viñedos se realizan de manera manual o semiautomática, lo cual no solo implica un alto consumo de tiempo y recursos, sino que también puede conllevar errores de estimación y dificultades en la gestión de grandes extensiones de terreno. La posibilidad de extraer automáticamente información estructural y espacial de los viñedos a partir de imágenes satelitales permitiría optimizar procesos clave como:

- La planificación del riego en función de la densidad real del cultivo.
- La estimación de rendimientos según la longitud y número de filas.
- La detección de anomalías o interrupciones en la estructura del viñedo.
- La monitorización histórica y comparativa de parcelas sin intervención *in situ*.

Además del impacto económico, estas mejoras tienen una dimensión medioambiental relevante, ya que permiten un uso más eficiente de recursos hídricos y fertilizantes, al ajustarse a la realidad estructural del viñedo y no a estimaciones genéricas.

Por ejemplo, en explotaciones de gran tamaño donde no se puede realizar una inspección visual frecuente, un sistema automático de análisis estructural permitiría identificar áreas de baja densidad de plantación, indicando la necesidad de replantación o intervención localizada. Asimismo, en zonas con restricciones hídricas, ajustar el riego al patrón real de plantación puede representar una diferencia significativa en el consumo de agua a lo largo de una campaña.

Sin embargo, el análisis de este tipo de imágenes presenta desafíos técnicos significativos: la variabilidad en las condiciones de iluminación, la resolución limitada, la presencia de cultivos vecinos o elementos ajenos (caminos, construcciones) y la heterogeneidad en la orientación y densidad de las filas dificultan la automatización del proceso.

En este contexto, el desarrollo de modelos de segmentación semántica basados en redes neuronales convolucionales, como *U-Net*, ofrece una solución robusta y escalable que puede

adaptarse a distintas condiciones y proporcionar resultados precisos en la identificación y análisis estructural de viñedos. Este trabajo se plantea como una contribución en esta línea, explorando y validando un modelo específico entrenado con imágenes satelitales y orientado a la generación automática de métricas agronómicas de interés.

1.2. Objetivos

El objetivo principal de este proyecto es desarrollar un sistema de inteligencia artificial capaz de identificar, contar y analizar las filas de viñedos a partir de imágenes satelitales. A partir de dicha segmentación, se pretende calcular métricas agronómicas relevantes como:

- La longitud total y media de las filas de viñedo.
- Estimación del número de cepas.
- Productividad esperada.

Para alcanzar este objetivo principal, se establecen los siguientes objetivos específicos:

- **OBJ-01** Investigación en modelos de inteligencia artificial para segmentación.
- **OBJ-02** Adquisición y preprocesamiento de imágenes satelitales de zonas vitícolas.
- **OBJ-03** Diseño y entrenamiento del modelo de segmentación U-Net.
- **OBJ-04** Posprocesamiento y análisis geométrico que permitan obtener métricas cuantificables.

1.3. Condicionantes

Para garantizar que el sistema desarrollado sea realista, útil y aplicable en escenarios concretos, se han identificado una serie de condicionantes. Estos se dividen en tres categorías:

1.3.1. Reglas de Negocio

Estas reglas se refieren a condiciones funcionales impuestas por el entorno de aplicación o los posibles usuarios del sistema:

- **RN-01** El sistema debe ser capaz de procesar imágenes satelitales de distintas fuentes y con variabilidad en la resolución, siempre que esta sea suficiente para identificar estructuras lineales.

- **RN-02** Las métricas generadas por el sistema deben ser interpretables por técnicos agrónomos y aportar un valor añadido a la toma de decisiones.
- **RN-03** Se prioriza la generalización del modelo, de modo que no se limite a un área geográfica específica ni dependa exclusivamente de un proveedor de imágenes.

1.3.2. Restricciones

Se trata de limitaciones tecnológicas, logísticas o metodológicas que condicionan el diseño y la implementación del proyecto:

- **R-01** Acceso limitado a conjuntos de datos etiquetados con precisión para el entrenamiento supervisado del modelo.
- **R-02** Recursos computacionales limitados para el entrenamiento del modelo (GPU de capacidad moderada), lo que implica restricciones en el tamaño del modelo y los tiempos de entrenamiento.
- **R-03** Tiempo acotado para el desarrollo del proyecto (plazo del TFG) lo cual condiciona la profundidad del análisis y la amplitud de los experimentos realizados.

1.3.3. Suposiciones

Son premisas que se asumen como ciertas durante el desarrollo del proyecto, y que condicionan el diseño de la solución:

- **S-01** Las filas de viñedos tienen una disposición lineal relativamente regular y diferenciable respecto al fondo o suelo circundante.
- **S-02** Las imágenes empleadas se han tomado en periodos de actividad vegetativa, lo que facilita la identificación visual de las plantas.
- **S-03** El modelo *U-Net*, debidamente entrenado, será capaz de generalizar adecuadamente a diferentes zonas geográficas con viñedos.

1.4. Alcance del proyecto

El presente proyecto se centra exclusivamente en el análisis de viñedos visibles en imágenes satelitales. No se contempla la detección de enfermedades o el análisis del estado vegetativo. Tampoco se desarrollarán interfaces de usuario ni aplicaciones móviles (aunque se integrará con la aplicación de un compañero, la cual ha sido desarrollada para su TFG); el alcance se limita al desarrollo del modelo, análisis de métricas y presentación de resultados en entorno técnico y académico. La metodología podrá ser extendida en trabajos futuros a otras aplicaciones agrícolas.

1.5. Estructura de la memoria

La presente memoria se estructura en dos grandes bloques, organizados en capítulos que siguen un orden lógico y metodológico:

1. **Descripción del proyecto**

   Este capítulo proporciona el contexto teórico y organizativo del trabajo, incluyendo:

   - **Introducción:** se plantea el problema al que quiere dar solución este TFG junto con los objetivos, condicionantes, alcance y estructura de la memoria.
   - **Planificación del proyecto:** incluye la metodología de trabajo empleada para la realización del proyecto, la gestión de riesgos y una estimación de los recursos y presupuesto necesario.
   - **Antecedentes:** habla del contexto teórico y técnico que rodea a la inteligencia artificial y revisa el estado del arte en técnicas de segmentación de imágenes aplicadas a la agricultura, prestando especial atención a la arquitectura *U-Net*.

2. **Desarrollo del proyecto**

   Este capítulo aborda de forma técnica el núcleo del trabajo, incluyendo:

   - **Desarrollo de la solución y experimentación**: se expone el proceso de entrenamiento y generación de predicciones del modelo *U-Net* con codificador *ResNet*, detallando cada una de sus fases. Además, se describe el procedimiento de ajuste de hiperparámetros llevado a cabo con el objetivo de encontrar la combinación óptima que minimice la función de pérdida durante el entrenamiento del modelo.
   - **Evaluación**: presenta las métricas obtenidas, análisis de errores, mejoras potenciales y comparativa con segmentaciones manuales.
   - **Conclusiones y trabajo futuro**: en este capítulo se presenta un balance del desarrollo realizado, abordando tanto los aspectos técnicos como la experiencia personal adquirida a lo largo del proyecto.

3. **Apéndices**

   - **Apéndice A**: Manual de Instalación
   - **Apéndice B**: Contenido adjunto

Capítulo 2

Planificación

Este capítulo detalla el enfoque metodológico adoptado para el desarrollo del proyecto, así como la planificación temporal, los recursos asignados, la gestión de riesgos y un balance final de tiempo y costes. La planificación del proyecto se ha guiado por un enfoque iterativo e incremental, basado en principios del desarrollo ágil. Este modelo de trabajo ha permitido una mejora continua del sistema mediante la entrega progresiva de resultados parciales, revisiones periódicas y adaptaciones constantes basadas en retroalimentación.

2.1. Metodología de trabajo

El desarrollo del sistema se ha organizado en iteraciones cortas llamadas *sprints*, con una duración aproximada de dos semanas cada una. Esta metodología ha facilitado la experimentación con diferentes modelos, el análisis continuo de resultados y una toma de decisiones informada a lo largo del proyecto.

La filosofía *agile* favorece una alta capacidad de respuesta al cambio, algo esencial en un proyecto de investigación aplicada como este, donde las decisiones técnicas evolucionan a partir del conocimiento que se va adquiriendo. En este marco, se ha seguido una aproximación cercana a *Scrum*, adaptada a un contexto unipersonal como es el de un Trabajo de Fin de Grado. Su principal objetivo es la consecución de los objetivos definidos en el proyecto, asegurando durante el proceso la implicación y comunicación de todos los agentes implicados a fin de maximizar la calidad del producto construido. Podemos diferenciar objetivos de aprendizaje que estructuran de forma genérica el trabajo necesario para completar el TFG:

- **Proyecto**: se centra en definir el planteamiento del problema y planificar de forma estructurada las actividades necesarias para su ejecución. Se adopta un enfoque iterativo e incremental, que permite adaptar el ritmo de trabajo y el progreso del estudiante al marco temporal establecido para la realización del Trabajo de Fin de Grado (TFG).
- **Antecedentes**: comprende todas las tareas orientadas a la comprensión profunda del contexto en el que se desarrolla el proyecto. Esto implica asimilar tanto el entorno

de aplicación (por ejemplo, el sector agrícola, industrial o social) como los aspectos científico-técnicos relevantes. Resulta esencial para identificar necesidades específicas, conocer soluciones ya existentes y adquirir una visión global de los conceptos que fundamentan el desarrollo del proyecto.

- **Desarrollo**: se lleva a cabo la construcción efectiva del sistema o producto. Las actividades realizadas aquí varían según la naturaleza del proyecto (ya sea de investigación, de desarrollo o mixto) e incluyen desde el diseño técnico hasta la implementación y pruebas de los componentes clave.

- **Aceptación**: tiene como objetivo verificar que el resultado obtenido cumple con los objetivos planteados inicialmente. Incluye un análisis crítico por parte del estudiante sobre las técnicas utilizadas, valorando su idoneidad y eficacia en relación con los resultados alcanzados.

- **Comunicación**: engloba dos componentes fundamentales: la elaboración de la memoria técnica y la defensa oral del proyecto. La memoria recoge de forma ordenada los aprendizajes, decisiones y conclusiones obtenidas a lo largo del proceso. La defensa, por su parte, permite exponer el trabajo realizado ante un tribunal evaluador. Ambos procesos se desarrollan de forma iterativa durante el proyecto, con el acompañamiento del tutor o tutores, lo que permite incorporar mejoras progresivamente y garantizar un resultado final de calidad.

2.1.1. Roles

La colaboración entre los distintos roles implicados (estudiante, tutor, comunidad académica y tribunal evaluador) genera una sinergia que permite un mayor control y seguimiento del proceso de desarrollo del proyecto, lo que se traduce en un producto final de mayor calidad y precisión.

- **Estudiante**: constituye la figura central del proyecto, siendo el principal responsable de ejecutar las tareas necesarias para alcanzar los objetivos establecidos. Su proceso de aprendizaje se enriquece mediante el *feedback* recibido de los demás actores. Además, debe asumir funciones de planificación y gestión, manteniendo actualizado tanto el tablero del proyecto como el cuaderno de trabajo, lo que contribuye a una mejor organización y trazabilidad.

- **Tutor**: desempeña un papel clave en el acompañamiento formativo del estudiante, ofreciendo orientación, apoyo técnico y retroalimentación continua. Su implicación es especialmente relevante en las fases iniciales del proyecto, donde colabora activamente en la definición del problema, la delimitación de objetivos y la identificación de recursos bibliográficos clave. A lo largo del proyecto, evalúa el progreso del estudiante y, si es necesario, ajusta los objetivos para alinearlos con el ritmo de avance.

- **Comunidad**: engloba a todas aquellas personas (compañeros, docentes o expertos) que pueden aportar valor al desarrollo del proyecto. La interacción con esta comunidad favorece un entorno colaborativo de aprendizaje en el que se comparten diferentes perspectivas y experiencias.

- **Tribunal**: su papel cobra protagonismo durante la fase final del proyecto, concretamente en el acto de defensa y evaluación. El tribunal valora tanto la calidad de los entregables como la presentación realizada por el estudiante, evaluando el grado de cumplimiento de los objetivos y la solidez de los conocimientos demostrados.

2.1.2. Eventos

La planificación temporal del proyecto se estructura mediante una secuencia de *sprints* de corta duración. En cada *sprint* se definen las historias de aprendizaje que deben completarse durante ese periodo, lo que permite establecer objetivos concretos, alcanzables y a corto plazo. Esta metodología facilita la posibilidad de redirigir y reformular dichos objetivos al comienzo de cada nuevo *sprint*, adaptándose así al progreso real del trabajo.

Con el fin de fomentar la mejora continua, al término de cada *sprint* se realiza una entrega parcial del proyecto, a partir de la cual se obtiene retroalimentación valiosa para ajustar el rumbo del desarrollo. Cada *sprint* incluye, además, una serie de eventos específicos que marcan y organizan su evolución.

- **Reunión de inicio:** Marca el comienzo del *sprint*. En esta reunión, el tutor define inicialmente el alcance del trabajo previsto, que posteriormente se consensúa con el estudiante para establecer la carga de trabajo. Esto incluye la determinación de los objetivos e historias de aprendizaje que se abordarán. A partir de ahí, corresponde al estudiante descomponer dichas historias en tareas específicas, asignándoles una duración estimada.

- **Reunión de sincronización:** Se lleva a cabo semanalmente y tiene una duración aproximada de quince minutos. En este encuentro, el estudiante informa al tutor sobre los avances logrados durante la semana anterior y expone las tareas que planea realizar en la siguiente. Además, es un espacio para plantear dudas, compartir posibles bloqueos y recibir orientación para resolverlos.

- **Comunicación de progresos:** Tiene lugar al finalizar cada *sprint*. En esta sesión, el estudiante realiza una presentación del estado actual del proyecto, mostrando el producto parcial desarrollado hasta ese momento.

- **Retrospectiva:** Constituye el evento final del *sprint*. En él, se realiza una evaluación anónima donde se identifican los aspectos positivos y negativos del *sprint*, así como posibles acciones de mejora. Este ejercicio fomenta la reflexión crítica y la mejora continua del proceso.

2.1.3. Artefactos

En el marco metodológico adoptado para el desarrollo del proyecto, los artefactos juegan un papel central como evidencias del progreso y como herramientas de mejora continua. Entre estos artefactos destacan especialmente el incremento y la retroalimentación:

- **Incremento:** Cada *sprint* culmina con un conjunto de entregables que constituyen el incremento, es decir, la representación tangible del trabajo realizado hasta ese momento. Este producto parcial refleja la consolidación de los avances alcanzados y sirve como base para continuar desarrollando nuevas funcionalidades o mejoras.
- **Retroalimentación:** Una vez finalizado el *sprint*, el estudiante recibe correcciones y comentarios por parte del tutor, quien evalúa el trabajo entregado. Gracias a estas intervenciones, el estudiante puede identificar posibles errores, confirmar enfoques acertados y encontrar nuevas oportunidades de mejora. En consecuencia, la retroalimentación se convierte en un pilar esencial para orientar los siguientes pasos del proyecto de forma informada y constructiva.

2.1.4. Entorno de trabajo

El entorno de trabajo ha sido principalmente digital, con las siguientes herramientas:

- ***Microsoft Teams***: para la comunicación con los tutores. *Microsoft Teams* ofrece varios servicios, como videoconferencias, calendario o mensajería instantánea, incluyendo el envío de imágenes u otro tipo de archivos.
- ***Trello***: tablero utilizado para facilitar la visualización del flujo de trabajo, permitiendo planificar y monitorizar el avance de los distintos objetivos, historias de aprendizaje y tareas específicas asociadas a cada *sprint*. Una de las herramientas más populares para implementar este tipo de tableros es *Trello*.

  Es un tablero de tipo Kanban, una herramienta visual ampliamente utilizada en metodologías ágiles para la gestión y seguimiento de tareas. Este tipo de tablero se estructura en tres columnas principales: la primera contiene las tareas pendientes de ejecución (*To Do*), la segunda agrupa aquellas que se encuentran actualmente en proceso (*Doing*), y la tercera recoge las tareas ya completadas (*Done*).
- **Cuaderno de trabajo**: herramienta destinada al registro sistemático del tiempo dedicado a cada una de las tareas planificadas. Su uso contribuye a establecer una rutina de trabajo estructurada, facilitando tanto el control del progreso como la autorregulación del estudiante a lo largo del proyecto.
- ***CVAT***: utilizada para anotar las imágenes del *dataset* empleado para entrenar el modelo. *CVAT* (*Computer Vision Annotation Tool*) es una herramienta web de código abierto diseñada para anotar datos de imagen y video, especialmente para

entrenar modelos de visión por computador, como los usados en detección de objetos, segmentación o clasificación. Esta herramienta ha sido fundamental en el trabajo ya que permite descargar máscaras de segmentación, es decir, archivos tipo PNG con las anotaciones realizadas de cada imagen que representan los lugares de la imagen donde está la vegetación que nos interesa. Otras herramientas similares como *Roboflow* permiten descargar archivos TXT o JSON, pero no PNG.

- ***Visual Studio Code y Jupyter Notebooks***: como entornos de desarrollo interactivos para pruebas y entrenamiento de modelos.

  Visual Studio Code (VS Code) es un editor de código fuente ligero, gratuito y multiplataforma, desarrollado por *Microsoft*, ampliamente utilizado por programadores y desarrolladores para escribir y editar código en una gran variedad de lenguajes de programación como Python, JavaScript, C++, Java, entre otros. Las principales características que han llevado a su elección es su facilidad de uso, el control de versiones integrado con *Git* y *GitHub*, las extensiones con las que cuenta o el terminal integrado que permite ejecutar comandos directamente desde el editor.

  Un archivo *Jupyter Notebook* es un archivo con extensión .ipynb (de *Interactive Python Notebook*). Este tipo de archivo contiene código fuente ejecutable (generalmente en Python, pero puede ser de otros lenguajes), texto descriptivo con formato Markdown, visualizaciones incrustadas (gráficos, imágenes, etc.) o resultados de ejecución, como salidas de código, tablas, errores, ... Son ideales para documentar paso a paso un análisis o experimento al poder combinar código, explicaciones y resultados en un único archivo.

- **Python**: base para todo el desarrollo, con bibliotecas como Torchvision, Matplotlib, NumPy, OpenCV o Scikit-learn.

- ***QGIS***: para visualización y análisis de imágenes satelitales y datos geoespaciales. *QGIS* (antes conocido como *Quantum GIS*) es un software libre y de código abierto para trabajar con información geográfica. Sirve para visualizar, editar, analizar y representar datos espaciales en mapas. Es una de las herramientas más potentes y populares dentro del ámbito de los sistemas de información geográfica (SIG o *GIS* en inglés).

- ***GitHub***: para control de versiones, almacenamiento y colaboración (aunque el proyecto sea unipersonal). *GitHub* es una plataforma web para almacenar, gestionar y compartir código de forma colaborativa. Se basa en *Git*, un sistema de control de versiones, y es ampliamente utilizada en desarrollo de software, ciencia de datos, inteligencia artificial y proyectos académicos.

- **LaTeX**: para la redacción de esta memoria. LaTeX, un sistema de preparación de documentos orientado a la creación de textos científicos, técnicos y académicos de alta calidad tipográfica, especialmente aquellos que contienen fórmulas matemáticas, gráficos, tablas o estructuras complejas. He seleccionado esta herramienta por los

resultados profesionales que proporciona (mejor que herramientas como *Microsoft Word* en estos casos) y su control total sobre el formato y la estructura. A través de Overleaf, que es una plataforma online para escribir, editar y compilar documentos en LaTeX, sin necesidad de instalar nada en tu ordenador, se ha escrito la memoria. Esta herramienta permite escribir documentos y ver el resultado (en formato PDF) en tiempo real, además de poder compartir los documentos con otras personas para una colaboración fluida.

- ***Mendeley***: para la gestión de bibliografías y referencias. *Mendeley* facilita la gestión de bibliografías y citas y simplifica el proceso de creación de referencias, en este caso en formato bib. Su plataforma intuitiva y sus herramientas de organización lo convierten en una herramienta muy útil en el sector de la investigación.

- ***ChatGPT***: utilizado como asistente de programación y generación de texto. *ChatGPT* es un modelo de lenguaje desarrollado por *OpenAI*, basado en la arquitectura GPT (*Generative Pre-trained Transformer*). Utiliza técnicas de aprendizaje profundo para comprender y generar texto de manera coherente y contextual. *ChatGPT* puede responder preguntas, redactar textos, traducir idiomas y asistir en la generación de contenido, lo que lo convierte en una herramienta valiosa para tareas de procesamiento de lenguaje natural.

Este conjunto de herramientas ha permitido un flujo de trabajo eficiente, reproducible y profesional. Además, se ha utilizado una máquina virtual de la universidad para llevar a cabo la ejecución de los distintos *scripts* utilizados para el entrenamiento de los modelos y otras funciones al contar con una capacidad de computación mayor que la de cualquier ordenador o portátil al alcance y de forma gratuita.

2.2. Planificación temporal

El proyecto se ha dividido en cuatro *sprints*, cada uno con objetivos específicos y duración aproximada de dos semanas, comenzando el 28 de Abril y finalizando el 22 de Junio. Aunque cada *sprint* tiene, como hemos dicho, objetivos específicos, todos comparten algunos como la planificación de los *sprints* y la documentación o redacción de la memoria, estos objetivos se incluirán más abajo en el diagrama de Gantt 2.1 por separado, para no repetir los objetivos en cada *sprint*.

Para estimar el tiempo que llevará finalizar el proyecto se usará la técnica *PERT (Program Evaluation and Review Technique)*. La duración de las tareas se considera una variable aleatoria con distribución Beta, y aproximaremos el tiempo esperado (TE) a partir de los valores T_o, T_m y T_p.

- T_o es el tiempo optimista, suponiendo que no se presenta ningún problema.

- T_m el tiempo más probable, bajo condiciones normales.

- T_p es el tiempo pesimista, donde ocurren contratiempos que retrasarían el tiempo de finalización de la tarea.

- TE es el tiempo esperado, la esperanza media de la variable aleatoria, que se calcula de la siguiente manera:

$$TE = \frac{T_o + 4T_m + T_p}{6}$$

  Con desviación típica

$$\sigma = \frac{T_o - T_p}{6}$$

2.2.1. Sprint #1

28 de Abril - 11 de Mayo

En este primer *sprint*, el objetivo principal fue adquirir una comprensión sólida del dominio del problema y de las tecnologías existentes:

- Investigación sobre segmentación semántica en agricultura de precisión.

- Exploración de arquitecturas de redes neuronales convolucionales para segmentación (*YOLO, U-Net, DeepLab*, ...).

- Selección y anotación de *datasets* relevantes.

Este *sprint* sentó las bases técnicas y conceptuales sobre las que se construiría el resto del proyecto.

Tarea	$\mathbf{T_o}$	$\mathbf{T_m}$	$\mathbf{T_p}$	**TE**
Investigación sobre segmentación semántica	14	18	22	18
Exploración de arquitecturas	20	23	26	23
Selección y anotación de *datasets*	26	28	30	28
Planificación	1	2	3	2
Redacción de la memoria	3	4	5	4

Tabla 2.1: Estimación del tiempo necesario para realizar las tareas Sprint #1

2.2.2. Sprint #2

12 de Mayo - 25 de Mayo

El segundo *sprint* se centró en la prueba de distintas arquitecturas y configuraciones:

- Implementación de prototipos con arquitecturas como *YOLOv11, ResUNet y DeepLabV3+*.
- Entrenamiento con subconjuntos de datos para evaluar comportamiento y eficiencia.
- Análisis crítico de los resultados preliminares.
- Anotación del *dataset* ampliado.

Este *sprint* fue crucial para reducir la incertidumbre y fundamentar la elección del modelo definitivo.

Tarea	$\mathbf{T_o}$	$\mathbf{T_m}$	$\mathbf{T_p}$	**TE**
Implementación de prototipos	28	32	36	32
Entrenamiento con subconjuntos de datos	10	12	14	12
Análisis crítico de los resultados preliminares	6	7	8	7
Anotación del *dataset* ampliado	16	18	20	18
Planificación	1	2	3	2
Redacción de la memoria	3	4	5	4

Tabla 2.2: Estimación del tiempo necesario para realizar las tareas Sprint #2

2.2.3. Sprint #3

26 de Mayo - 8 de Junio

Con el modelo elegido, el tercer *sprint* se dedicó a su desarrollo completo:

- Entrenamiento del modelo *U-Net* sobre el *dataset* completo.
- Ajuste de hiperparámetros mediante validación cruzada.
- Evaluación de resultados con métricas como IoU o precisión.

Este *sprint* produjo un sistema funcional capaz de segmentar con precisión las filas de viñedos en imágenes satelitales.

Tarea	$\mathbf{T_o}$	$\mathbf{T_m}$	$\mathbf{T_p}$	**TE**
Entrenamiento del modelo *U-Net*	15	20	25	20
Ajuste de hiperparámetros	17	20	23	20
Evaluación de resultados	20	26	32	26
Planificación	2	3	4	3
Redacción de la memoria	4	6	8	6

Tabla 2.3: Estimación del tiempo necesario para realizar las tareas Sprint #3

2.2.4. Sprint #4

9 de Junio - 22 de Junio

El *sprint* final se centró en la evaluación rigurosa del sistema y en la generación de productos finales:

- Validación cruzada sobre diferentes regiones.
- Extracción de métricas agronómicas como densidad de plantación, longitud de filas y cobertura vegetal.
- Preparación de informes, gráficos y visualizaciones para la memoria.
- Elaboración de conclusiones y posibles mejoras futuras.

Tarea	$\mathbf{T_o}$	$\mathbf{T_m}$	$\mathbf{T_p}$	**TE**
Validación cruzada	12	15	18	15
Extracción de métricas agronómicas	16	20	24	20
Preparación de informes, gráficos y visualizaciones	6	10	14	10
Elaboración de conclusiones y posibles mejoras	4	5	6	5
Planificación	3	5	7	5
Redacción de la memoria	19	20	21	20

Tabla 2.4: Estimación del tiempo necesario para realizar las tareas Sprint #4

El correspondiente diagrama de Gantt permite observar de forma gráfica la distribución temporal de las tareas. El eje X representa las semanas, es decir, cada rectángulo coloreado corresponde a una semana.

La duración de las tareas es aproximada, para facilitar su lectura en el diagrama y no tener que partirlo en varios diagramas de Gantt más pequeños y subdivididos. Los colores simbolizan los distintos objetivos y las tareas que los componen.

Como se ha comentado antes, las tareas de planificación y redacción de la memoria, en la parte baja del diagrama, aparecen durante una gran parte de la duración del proyecto, ya que se deben realizar de forma complementaria al resto de tareas.

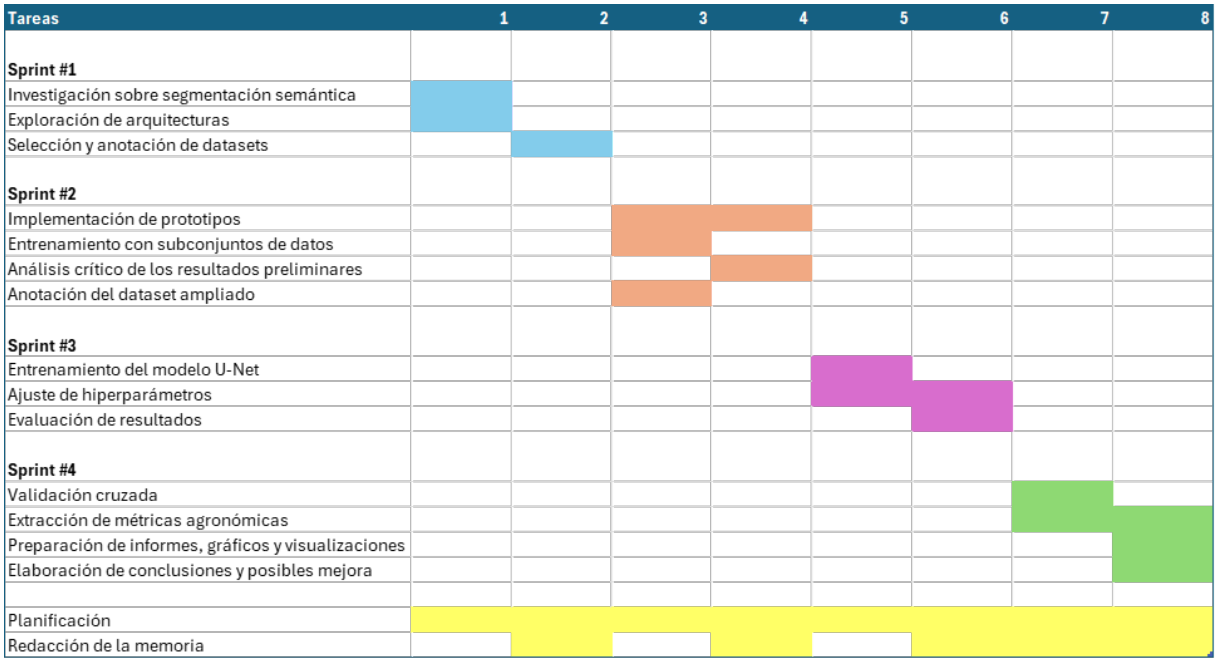


Figura 2.1: Diagrama de Gantt

2.3. Presupuestos

Para la estimación del presupuesto se tendrán en cuenta las herramientas utilizadas (*hardware* y *software*, con los factores de impacto que correspondan a la duración del proyecto), junto con los recursos humanos necesarios, según la planificación de tareas, y el tipo de rol correspondiente a cada tarea.

- ***Hardware***: el trabajo ha sido realizado en un ordenador portátil, modelo Lenovo IdeaPad S145-15IWL. Posee una CPU Intel Core i7-8565U y 8 GB de memoria RAM. Su precio es 799 €. Se estima una vida útil de 5 años y una duración del proyecto de 2 meses, por tanto, el coste del uso ha sido de

$$799\ € \cdot \frac{2\ \text{meses}}{5 \cdot 12\ \text{meses}} = 26,63\ €$$

Por otro lado se ha usado una máquina virtual de la universidad, cuyo coste aproximado estimamos en 30 €, por estos 2 meses.

Por tanto, el presupuesto aproximado por parte del *hardware* ha sido 56,63 €.

- ***Software* y licencias**: todas las herramientas utilizadas son gratuitas de código abierto, por lo que no tienen impacto en el presupuesto.

- **Recursos Humanos**: La asignación de recursos humanos en este proyecto contempla una duración total de 300 horas, siendo ejecutado íntegramente por una única persona que asume cuatro roles diferenciados a lo largo del desarrollo.

  El gestor de proyecto es responsable de la planificación y supervisión de las distintas tareas, con un salario bruto estimado de 10,77 €/hora. El rol de analista se centra en la interpretación, organización y manipulación de los datos, con una retribución de 14,36 €/hora.

  El perfil de científico de datos abarca el análisis avanzado de datos y la extracción de información relevante, empleando técnicas propias de la estadística y la inteligencia artificial; este rol está remunerado con 18,43 €/hora. Por último, el programador se ocupa del desarrollo del código y la implementación técnica de la solución, con un salario de 14,70 €/hora.

  Teniendo esto en cuenta, se ha estimado una dedicación de 12 horas asociadas al rol de gestor de proyecto, 89 horas para el rol de analista, 109 horas al rol de científico de datos y 90 horas al rol de programador.

  Aplicando estas tarifas a las horas estimadas por rol, el coste total de los recursos humanos implicados en el desarrollo del proyecto asciende a:

Puesto	**Salario por hora**	**Horas**	**Salario Total**
Gestor de Proyecto	10,77 €/h	12	129,24 €
Analista	14,36 €/h	89	1278,04 €
Científico de datos	18,43 €/h	109	2008,87 €
Programador	14,70 €/h	90	1323 €
Total		300	4739,15 €

Tabla 2.5: Presupuesto estimado empleado en Recursos Humanos

A este coste total es necesario añadir la cotización correspondiente a la Seguridad Social, derivada de la contratación de un trabajador. Esta supone un 28,30 % del salario bruto, según la legislación vigente. Por tanto, el coste final asociado a los

recursos humanos se calcularía incluyendo dicho porcentaje adicional sobre el salario bruto estimado:

$$P_{RRHH} = S_T + SS = 4739,15 + 4739,15 \cdot 28,30\,\% = 6080,33\,€$$

Presupuesto total estimado: Sumando los costes de los distintos elementos, se obtiene un presupuesto total de 6136,95 €.

Este presupuesto sirve como referencia para futuros desarrollos similares o para evaluar la escalabilidad del proyecto en un contexto profesional.

2.4. Gestión de riesgos

La identificación temprana y la planificación frente a posibles riesgos es clave para el éxito del proyecto. Permite anticipar problemas, mitigar su impacto y garantizar la continuidad del proyecto.

Para llevar a cabo este proceso, es fundamental identificar los posibles riesgos y realizar un análisis cualitativo que considere tanto su probabilidad de ocurrencia como su impacto potencial. Posteriormente, se debe efectuar un análisis cuantitativo que permita valorar numéricamente su alcance. A partir de estos análisis, se desarrollan planes de contingencia destinados a mitigar las amenazas identificadas. Finalmente, se establece un sistema de seguimiento y control para supervisar la evolución de los riesgos a lo largo del proyecto.

2.4.1. Identificación de riesgos

- **R01:** Escasez de datos etiquetados o calidad insuficiente.
- **R02:** Complejidad del problema mayor a la prevista.
- **R03:** Limitaciones computacionales en etapas de entrenamiento.
- **R04:** Retrasos en la planificación por sobrecarga académica.

2.4.2. Estimación de los riesgos

Las siguientes tablas presentan un análisis de la probabilidad de ocurrencia de cada riesgo y del impacto asociado en caso de que dicho riesgo se materialice. Ambos factores se cuantifican utilizando una escala que varía en el rango de [0, 5].

Riesgo	Probabilidad	Justificación
R01	4	Debido a la naturaleza específica del problema, existen pocos conjuntos de datos públicos disponibles, o directamente no existen.

R02	3	Solo una asignatura del grado trata conceptos relacionados con los sistemas inteligentes, por lo que es habitual que no se pueda profundizar en todos los aspectos de este campo.
R03	3	Es muy habitual que las técnicas utilizadas demanden una mayor capacidad computacional para poder ejecutarse en un tiempo razonablemente corto.
R04	2	Es razonable asumir que, a lo largo de un curso académico, puedan producirse retrasos debido a la realización de exámenes o la entrega de trabajos.

Tabla 2.6: Probabilidad de cada riesgo

Riesgo	**Impacto**	**Justificación**
R01	5	No seria posible entrenar correctamente los modelos.
R02	3	Provocaría un retraso en el desarrollo del proyecto al necesitar más tiempo para solucionar la complejidad abordada.
R03	4	La ausencia de una GPU con la capacidad necesaria puede suponer un impedimento significativo para el desarrollo eficiente del proyecto.
R04	3	Puede conllevar retrasos en la entrega del proyecto respecto a la fecha prevista.

Tabla 2.7: Impacto de cada riesgo

Después de calcular el producto Exposición = Probabilidad $\times$ Impacto, se obtienen distintos valores que permiten clasificar los riesgos según su prioridad:

- Prioridad alta: $10 \leq$ Exposición
- Prioridad media: $5 \leq$ Exposición ≤ 10
- Prioridad baja: $5 \leq$ Exposición

Riesgo	Probabilidad	Impacto	Exposición
R01	4	5	20
R02	3	3	9
R03	3	4	12
R04	2	3	6

Tabla 2.8: Matriz Probabilidad x Riesgo

2.4.3. Plan de contingencia

A continuación se detalla el plan de contingencia para cada uno de los riesgos mencionados:

Riesgo	Plan de contingencia
R01	Uso de *datasets* públicos alternativos.
R02	Simplificación del alcance si es necesario.
R03	Uso de servicios gratuitos o de bajo coste con GPU.
R04	Planificación flexible para acomodar cargas académicas.

Tabla 2.9: Plan de Contingencia

2.5. Balance

En esta sección se analizan las diferencias entre la planificación inicial y los resultados reales obtenidos al finalizar el proyecto.

2.5.1. Balance temporal

La siguientes tablas presentan una comparación entre el tiempo estimado y el tiempo efectivamente dedicado a cada tarea, en horas. En general, se observa una correspondencia razonable entre ambos valores, salvo en algunos casos concretos donde se produjeron desviaciones respecto a lo previsto.

En relación con la investigación sobre segmentación semántica en la agricultura, no existía información clara y relacionada con el problema que aquí planteamos, por lo que

tuve que abstraer un poco el problema a algo más general sobre segmentación. Esto hizo que la tarea excediera el tiempo estimado.

Por otro lado, la anotación de las imágenes dependía de la arquitectura o modelo que fuera a entrenar. Por ejemplo, *YOLO* utiliza archivos de tipo *.txt* mientras que *U-Net* utiliza máscaras o archivos *.png* para el entrenamiento. Esto hizo que la tarea llevara más tiempo del planeado, al tener que anotar las imágenes en distinto formato.

Respecto al entrenamiento de los modelos, el uso de diferentes librerías y formatos hizo que me llevara más tiempo del planificado, aunque los entrenamientos posteriores ya fueran más rápidos en cuanto a la no aparición de fallos en el código.

Por último, en cuanto a la preparación de informes y gráficos, el uso de librerías dedicadas a esta parte facilitó su elaboración, por lo que el tiempo empleado fue menor.

La diferencia final entre el tiempo estimado y el tiempo real en total fue de 4 horas, una diferencia despreciable y asumible en este contexto.

Tarea	**TE**	Tiempo Real
Investigación sobre segmentación semántica	18	23
Exploración de arquitecturas	23	24
Selección y anotación de *datasets*	28	32
Implementación de prototipos	32	31
Entrenamiento con subconjuntos de datos	12	13
Análisis crítico de los resultados preliminares	7	6
Anotación del *dataset* ampliado	18	18
Entrenamiento del modelo *U-Net*	20	18
Ajuste de hiperparámetros	20	21
Evaluación de resultados	26	23
Validación cruzada	15	16
Extracción de métricas agronómicas	20	23
Preparación de informes, gráficos y visualizaciones	10	6
Elaboración de conclusiones y posibles mejoras	5	4
Planificación	12	10
Redacción de la memoria	34	36
Tiempo Total	300	304

Tabla 2.10: Comparación entre el tiempo real y el tiempo estimado para cada tarea

2.5.2. Balance económico

El coste del proyecto se ve afectado directamente por la diferencia entre el tiempo estimado y el tiempo real. Dado que finalmente hemos empleado más tiempo del planeado, el coste del proyecto ascenderá. En la siguiente tabla se muestra una comparación entre el presupuesto estimado y el presupuesto real. El presupuesto real ha sido calculado teniendo en cuenta la diferencia de horas en los diferentes roles en el apartado de recursos humanos.

Puesto	**Salario por hora**	**Horas**	**Salario Total**
Gestor de Proyecto	10,77 €/h	15	161,55 €
Analista	14,36 €/h	86	1234,96 €
Científico de datos	18,43 €/h	115	2119,45 €
Programador	14,70 €/h	88	1293,6 €
Total		304	4809,56 €

Tabla 2.11: Presupuesto real empleado en Recursos Humanos

Como antes, añadiendo el coste de contratación:

$$P_{RRHH} = S_T + SS = 4809,56 + 4809,56 \cdot 28,30\,\% = 6170,66\ €$$

Concepto	**Estimado**	**Real**	**Diferencia**
Hardware	56,63 €	57,38 €	0,75 €
Software	0 €	0 €	0 €
RRHH	6080,33 €	6170,66 €	90,33 €
Total	6136,96 €	6228,04€	91,08 €

Tabla 2.12: Comparación entre el presupuesto estimado y real

Como se muestra en la tabla, la diferencia total es de 91,08 €, acorde a la diferencia de tiempo de la que hemos hablado anteriormente.

Capítulo 3

Antecedentes

Antes de abordar el desarrollo técnico de cualquier proyecto de ingeniería, es fundamental contextualizar adecuadamente su planteamiento. Este capítulo proporciona una visión completa de los antecedentes que sustentan el trabajo, desde la motivación económica y sectorial hasta los fundamentos teóricos y técnicos que hacen viable su desarrollo.

En primer lugar, se analiza el entorno de negocio en el que se enmarca el proyecto, abordando la relevancia del sector agrícola (y en particular de la viticultura) así como el papel creciente de la digitalización y el uso de tecnologías inteligentes en dicho ámbito. A continuación, el estado del arte recoge los trabajos y soluciones previas más representativas en la materia, permitiendo posicionar el presente trabajo en relación con ellos. Los fundamentos teóricos revisan los conceptos clave necesarios para entender las tecnologías empleadas, como la inteligencia artificial, el aprendizaje automático, las redes neuronales profundas y la segmentación de imágenes. Finalmente, se presentan los fundamentos técnicos del sistema, abordando las herramientas, lenguajes, librerías y *datasets* utilizados para su implementación.

3.1. Entorno de negocio

La viticultura representa una de las ramas más relevantes y con mayor tradición dentro del sector agroalimentario. Su importancia no solo radica en su contribución a la economía rural y al valor añadido de los productos derivados del vino, sino también en su papel como motor de desarrollo en muchas regiones productoras del mundo, incluyendo España, Francia, Italia, Chile, Argentina o Estados Unidos. A nivel mundial, el cultivo de la vid ocupa más de 7 millones de hectáreas y genera una actividad económica que supera los 300 mil millones de euros al año, según datos de la Organización Internacional de la Viña y el Vino (OIV).

En este contexto, la gestión eficiente del viñedo se ha convertido en una prioridad para los productores, especialmente ante los retos actuales: el cambio climático, el encarecimiento de los insumos agrícolas, la escasez de mano de obra cualificada, la necesidad de trazabilidad y sostenibilidad, y la creciente competencia en el mercado global. La presión para aumentar la productividad y reducir costes ha impulsado la adopción de nuevas

tecnologías, enmarcadas bajo el paradigma de la agricultura de precisión.

La agricultura de precisión consiste en aplicar tecnologías avanzadas, como sensores, sistemas de posicionamiento global (GPS), análisis de datos, inteligencia artificial (IA) e imágenes satelitales o de drones, con el fin de tomar decisiones informadas y específicas sobre la gestión de los cultivos. Este enfoque permite, entre otras cosas, optimizar el uso de fertilizantes y fitosanitarios, detectar enfermedades o estrés hídrico de forma temprana, y evaluar el rendimiento de las parcelas con alta resolución espacial y temporal.

Dentro de este marco, las imágenes satelitales se han posicionado como una herramienta clave para la monitorización remota de grandes extensiones agrícolas. Gracias a plataformas como *Sentinel-2* (Agencia Espacial Europea), *Landsat* (NASA/USGS) o incluso servicios comerciales como *PlanetScope*, hoy es posible acceder a datos multiespectrales de alta frecuencia y cobertura, lo que habilita múltiples aplicaciones para el seguimiento del estado fenológico, la detección de anomalías o la planificación de labores.

Sin embargo, para poder aprovechar todo el potencial de estas imágenes, es necesario desarrollar soluciones basadas en inteligencia artificial que permitan extraer información útil de forma automatizada y escalable. En particular, la segmentación de imágenes mediante redes neuronales convolucionales (CNNs) se ha convertido en una técnica de referencia para identificar objetos agrícolas (filas de cultivo, parcelas, árboles, etc.) dentro de imágenes de alta resolución.

Este proyecto se sitúa en la intersección de estos avances, proponiendo un sistema basado en redes *U-Net* (una arquitectura de segmentación ampliamente utilizada) para detectar, contar y analizar automáticamente las filas de viñedos a partir de imágenes satelitales. La información extraída puede ser usada para:

- Medir la longitud de las filas, lo que permite estimar el área cultivada de forma precisa.
- Calcular la densidad de plantación, clave para modelar el rendimiento potencial o detectar anomalías de distribución.
- Identificar patrones de plantación, lo cual puede indicar el tipo de manejo agronómico o la edad del viñedo.
- Detectar posibles errores o zonas sin cultivo, lo que permite mejorar la eficiencia de uso del terreno.

Además, la posibilidad de aplicar este sistema sobre imágenes históricas y multitemporales habilita análisis evolutivos y retrospectivos, útiles tanto para productores como para entidades reguladoras, compañías aseguradoras, empresas de análisis territorial y administraciones públicas.

En definitiva, este proyecto aporta valor en un entorno de negocio en transformación, donde la digitalización del sector vitivinícola se perfila como un elemento estratégico para mejorar la competitividad, la sostenibilidad y la resiliencia del cultivo de la vid.

3.2. Estado del arte

El análisis del estado del arte permite situar el trabajo desarrollado en el marco de investigaciones previas y tecnologías existentes, evaluando las distintas metodologías utilizadas para abordar problemáticas similares. En este caso, el foco se centra en los avances relacionados con la segmentación semántica de cultivos mediante técnicas de inteligencia artificial, especialmente en el contexto del análisis automatizado de viñedos a partir de imágenes satelitales.

Esta sección se estructura en dos partes. En primer lugar, se presenta una descripción de trabajos relacionados que abordan tareas similares, tanto desde el punto de vista técnico (modelos utilizados, fuentes de datos, métricas) como aplicado (casos reales en entornos agrícolas, especialmente viñedos). A continuación, se ofrece una discusión crítica, donde se analizan los principales enfoques encontrados, se identifican tendencias comunes, carencias y oportunidades de mejora, y se justifica la elección metodológica seguida en este proyecto.

El conocimiento de los antecedentes y desarrollos actuales no solo aporta contexto, sino que permite construir una solución mejor fundamentada, con un mayor potencial de aplicación y replicabilidad en escenarios reales.

3.2.1. Descripción de trabajos relacionados

El uso de imágenes satelitales y técnicas de inteligencia artificial para el análisis de cultivos ha experimentado un crecimiento notable en los últimos años, en particular en el marco de la agricultura de precisión. Uno de los enfoques más prometedores es la segmentación semántica mediante redes neuronales convolucionales, que permite identificar y clasificar objetos o áreas de interés dentro de imágenes de alta resolución. Entre las múltiples aplicaciones en el ámbito agrícola, la detección de filas de cultivo, y en particular de filas de viñedos, ha atraído especial atención debido a su estructura repetitiva y su relevancia productiva.

Segmentación de cultivos mediante IA

Uno de los modelos más utilizados para segmentación en el ámbito agrícola es la *U-Net*, una arquitectura de red neuronal convolucional propuesta inicialmente para segmentación biomédica, que ha demostrado un rendimiento sobresaliente en tareas donde se dispone de pocas imágenes etiquetadas. Gracias a su diseño en forma de "U", permite combinar eficazmente la información espacial y contextual, algo fundamental en imágenes satelitales.

Por ejemplo, Kattenborn et al. (2021) [27] desarrollaron un sistema basado en *U-Net* para clasificar especies de plantas individuales en imágenes aéreas hiperespectrales. En su estudio se destacó la capacidad del modelo para generalizar sobre diferentes regiones geográficas, aspecto clave en escenarios con variabilidad topográfica o climática.

Otro trabajo representativo es el de Wang et al. (2024) [61], que utilizaron un modelo *DeepLabv3+* para identificar cultivos y estructuras agrícolas en imágenes de satélite *Sentinel-2*. Aunque la resolución espacial de estas imágenes (10-20 m por píxel) limita el análisis de estructuras finas, como filas individuales, sí permite una clasificación precisa a

nivel de parcela.

Recientemente, se han desarrollado arquitecturas más eficientes y especializadas, como *ResLMFFNet* (2024), que mejora la precisión en segmentación en tiempo real sin incrementar el tamaño del modelo, mostrando gran potencial para aplicaciones agrícolas con datos UAV y satelitales. Además, trabajos como los de Irina Korotkova y Natalia Efremova (2023) [29] han comparado *U-Net, SegNet y DeepLabv3+* sobre imágenes *Sentinel-2*, confirmando la eficacia de variantes preentrenadas de *U-Net* para la segmentación en agricultura de precisión. Un estudio utilizó *U-Net* para segmentar campos de lavanda con un *dice* de 0,8324, demostrando eficacia incluso con bandas RGB de Sentinel-2.

En el ámbito de la segmentación de imágenes, las arquitecturas tradicionales como *FCN* o *U-Net* han sido ampliamente utilizadas debido a su simplicidad y eficacia. *U-Net*, en particular, se ha consolidado como un estándar en tareas de segmentación biomédica y agrícola, gracias a su diseño simétrico *encoder-decoder* y a su capacidad para recuperar detalles espaciales mediante conexiones de tipo *skip*.

***Transformers* en segmentación satelital**

En los últimos años han surgido enfoques más avanzados basados en mecanismos de atención (*Attention Mechanisms*) y *Transformers*, como *SegFormer*, *TransUNet* o *Swin-Unet*, que han mostrado resultados prometedores en *benchmarks* generales de segmentación semántica. Estos modelos destacan por su capacidad para modelar relaciones espaciales a largo plazo, lo cual puede ser especialmente útil en imágenes con estructuras repetitivas o contextos complejos.

Desde 2022, la adopción de arquitecturas basadas en *Transformers* ha comenzado a consolidarse en tareas de segmentación semántica, y los estudios de 2024 y 2025 reflejan una creciente aplicación de estas técnicas en el ámbito de la observación terrestre. Modelos como *SegFormer*, *Swin-Unet*, *TransUNet* y *Mask2Former* han sido objeto de evaluaciones exhaustivas en entornos agrícolas y forestales.

Particularmente, trabajos recientes han comenzado a explorar su aplicación directa sobre imágenes satelitales multitemporales o agrícolas. Por ejemplo:

- Gallo et al. (2024) [19] desarrollaron una variante de *Swin-UNETR* adaptada a series temporales satelitales para segmentar cultivos con alta precisión en regiones como Lombardía (Italia), obteniendo mejoras sustanciales respecto a *U-Net 3D* o *DeepLabV3*.

- MacDonald et al. (2024) [33] propusieron *VistaFormer*, un *Transformer* eficiente capaz de trabajar sobre imágenes *Sentinel-2* multitemporales. En sus experimentos, superó en precisión y eficiencia a modelos CNN tradicionales en tareas de segmentación agrícola.

Estos trabajos confirman que las arquitecturas *Transformer* están ganando relevancia también en escenarios de resolución media como los ofrecidos por plataformas satelitales, y abren nuevas vías para combinar segmentación con información temporal.

Además, estudios recientes también exploran el uso de modelos híbridos, como *U-Net Transformer Fusion* (UTF-Net), que combina un encoder basado en *Transformers* con un decoder clásico tipo U-Net, lo que permite aprovechar lo mejor de ambos mundos: capacidad de generalización y detalle espacial.

Aplicaciones específicas a viñedos

En el caso particular de los viñedos, Ayamga et al. (2023) [4] desarrollaron un enfoque basado en *U-Net* para segmentar viñedos en imágenes de drones en Sudáfrica. A pesar de que el enfoque estaba centrado en imágenes RGB de muy alta resolución, los resultados evidencian la capacidad de la red para identificar con precisión la disposición de las hileras.

También destaca el trabajo de Torres-Sánchez et al. (2015) [51], quienes utilizaron visión por computador y algoritmos de detección de líneas para identificar hileras de cultivo en imágenes aéreas tomadas por drones. Aunque no utilizaron redes neuronales, su enfoque por detección geométrica sentó las bases para posteriores mejoras mediante aprendizaje profundo.

En años recientes, Casado-García et al. (2023) [12] aplicaron *DeepLabv3-ResNeXt* para segmentar racimos de uva con alta precisión, demostrando robustez frente a variaciones de sensores y condiciones. Chiatti et al. (2023) [14] introdujeron técnicas de *fine-tuning* selectivo para adaptar modelos preentrenados a nuevos viñedos, optimizando la eficiencia y la capacidad de generalización.

Por último, iniciativas como la creación de *datasets* multiclase en regiones vitivinícolas italianas (2024) [48] avanzan en la disponibilidad de datos públicos para entrenar y evaluar modelos de segmentación, cubriendo viñedos, olivares y otros cultivos en hileras, un aporte relevante para la comunidad científica.

Proyectos comerciales como *VineView* o *Agrospace* ofrecen servicios basados en imágenes satelitales y algoritmos de IA para monitorización de viñedos, pero sus tecnologías son propietarias y no siempre están respaldadas por publicaciones académicas detalladas.

3.2.2. Discusión

El análisis de los trabajos relacionados permite identificar varias tendencias clave:

- **Dominancia de la arquitectura *U-Net***: La *U-Net* se ha consolidado como la arquitectura base para segmentación en el sector agrícola, especialmente en escenarios donde el etiquetado es costoso y limitado, como es el caso de los viñedos.
- **Uso creciente de imágenes de alta resolución**: Mientras que las imágenes de satélite como *Sentinel-2* permiten monitoreo de parcelas, para tareas más detalladas (como detección de filas) son preferibles imágenes de mayor resolución, obtenidas por drones o plataformas comerciales. Sin embargo, esto también limita la escalabilidad y frecuencia de actualización.
- **Limitaciones en *datasets* públicos**: Una de las principales barreras es la falta de conjuntos de datos públicos bien etiquetados para tareas específicas como la

segmentación de viñedos. Esto obliga a muchos investigadores a generar sus propios *datasets*, dificultando la comparación entre métodos y la reproducibilidad.

- **Transferibilidad geográfica**: Los modelos entrenados en una región no siempre generalizan bien a otras debido a variaciones en tipo de suelo, densidad de plantación, orientación de las hileras, climatología o incluso estilos de poda. Técnicas recientes de fine-tuning y validación cruzada regional mejoran la robustez y adaptabilidad.

- **Poca presencia de soluciones satelitales en resolución media para tareas finas**: Si bien las imágenes satelitales son adecuadas para tareas de clasificación general, su resolución puede resultar insuficiente para segmentación de elementos pequeños, como filas individuales. Esto abre un espacio de innovación para modelos optimizados capaces de trabajar con resolución limitada y aun así obtener resultados precisos.

- **Necesidad de soluciones abiertas y reproducibles**: Gran parte de los estudios y soluciones comerciales no publican su código ni sus *datasets*, lo que limita la reproducibilidad y transferencia del conocimiento. La comunidad científica se está movilizando cada vez más hacia el desarrollo de soluciones abiertas.

- **Emergencia de *Transformers* en agricultura**: Aunque todavía no son la opción por defecto, los *Transformers* están demostrando una mayor capacidad para segmentar estructuras complejas en imágenes satelitales multitemporales, especialmente cuando se dispone de datos suficientes y capacidad computacional.

En este contexto, el presente proyecto contribuye con una propuesta concreta y reproducible para la detección automática de filas de viñedo mediante segmentación semántica con *U-Net*, utilizando imágenes satelitales accesibles como las de *Sentinel-2* o plataformas equivalentes. Aporta valor al enfocarse en una problemática agrícola concreta, desarrollando una solución adaptable, escalable y con potencial de generalización a otros cultivos en hileras (olivares, frutales, etc.), respaldada por las tendencias y avances recientes.

Aunque los *Transformers* ofrecen ventajas teóricas, su alto coste en términos de recursos, la necesidad de grandes volúmenes de datos para entrenamientos efectivos, y su complejidad arquitectónica hacen que, en este contexto específico de viñedos con imágenes limitadas, una solución basada en *U-Net* resulte más adecuada.

Se podría ampliar en futuras iteraciones el análisis comparativo con estos modelos más recientes, pero la elección actual está alineada con los objetivos de simplicidad, reproducibilidad y aplicabilidad práctica inmediata del sistema desarrollado.

3.3. Fundamentos teóricos

Para comprender y justificar las decisiones técnicas adoptadas en este proyecto, es imprescindible contar con una base sólida de conocimientos sobre las disciplinas que lo

sustentan. En particular, el sistema desarrollado se basa en técnicas avanzadas de inteligencia artificial y aprendizaje automático, con un enfoque centrado en redes neuronales profundas para la segmentación semántica de imágenes satelitales.

Esta sección presenta una revisión de los conceptos fundamentales que hacen posible el diseño y la implementación de este tipo de sistemas. Se abordan, de forma progresiva, los pilares teóricos de la inteligencia artificial (IA), el aprendizaje automático (*Machine Learning*), las redes neuronales, el aprendizaje profundo (*Deep Learning*) y las técnicas de segmentación de imágenes, prestando especial atención al modelo *U-Net*, que constituye el núcleo del sistema desarrollado.

3.3.1. Inteligencia Artificial

La inteligencia artificial es una de las ramas más influyentes de la informática moderna y uno de los pilares tecnológicos del siglo XXI. Su propósito es crear sistemas capaces de replicar, y en ocasiones superar, ciertas capacidades cognitivas humanas, como el razonamiento, el aprendizaje, la percepción visual, la comprensión del lenguaje o la toma de decisiones. Lejos de ser un campo meramente teórico, la IA ha demostrado su aplicabilidad en un amplio rango de dominios, desde el diagnóstico médico y la conducción autónoma hasta el análisis automático de cultivos agrícolas, como se aborda en este proyecto.

En el contexto específico del TFG, la IA se emplea para desarrollar un sistema inteligente capaz de segmentar e interpretar imágenes satelitales con el objetivo de identificar y analizar filas de viñedos. Esta tarea, que requiere reconocimiento visual complejo, forma parte del área de visión artificial, una de las aplicaciones más relevantes de la IA moderna.

Los orígenes de la IA se remontan a mediados del siglo XX, cuando se empezó a investigar formalmente sobre la posibilidad de construir "máquinas inteligentes". Uno de los hitos más influyentes fue la publicación, en 1950, del artículo *"Computing Machinery and Intelligence"* por Alan Turing [52], donde propuso el famoso Test de Turing como criterio para determinar si una máquina puede exhibir un comportamiento inteligente indistinguible del humano.

La conferencia de Dartmouth en 1956, organizada por John McCarthy, Marvin Minsky y otros pioneros, se considera el acto fundacional de la IA como campo de estudio. Desde entonces, la evolución de la inteligencia artificial ha atravesado diversas etapas:

- **Décadas de 1950–1970**: se desarrollaron sistemas basados en reglas y lógica formal. Aunque exitosos en dominios muy acotados, eran frágiles y no escalaban bien.
- **Décadas de 1980–1990**: se introdujo el enfoque de sistemas expertos, con bases de conocimiento y motores de inferencia. También se empezaron a estudiar redes neuronales, aunque su uso era limitado por la falta de potencia computacional.
- **Desde los 2000**: el acceso a grandes cantidades de datos (*big data*), la mejora en el hardware (particularmente GPUs) y la innovación en algoritmos impulsaron una nueva etapa: la del aprendizaje automático (*machine learning*) y, posteriormente, el aprendizaje profundo (*deep learning*).

- **A partir de 2012**: con el éxito de modelos como *AlexNet*, basados en redes neuronales convolucionales, la IA experimenta una expansión global con aplicaciones cada vez más sofisticadas y generalizadas.

De forma muy general, se puede decir que la IA busca construir entidades inteligentes. Sin embargo, a lo largo del tiempo, se han propuesto diferentes definiciones y enfoques para conceptualizar la inteligencia artificial. Estos pueden clasificarse en cuatro grandes corrientes, según se enfoque en el pensamiento o el comportamiento, y en si se toma como referencia al ser humano o a un ideal racional:

Tipo de IA	**Enfoque**	**Objetivo Principal**
Pensar como humanos	Enfoque cognitivo	Imitar procesos mentales humanos (psicología, neurociencia)
Actuar como humanos	Enfoque conductual	Simular comportamiento humano (Test de Turing)
Pensar racionalmente	Enfoque lógico	Seguir reglas de inferencia lógica
Actuar racionalmente	Enfoque funcional	Maximizar una medida de rendimiento (agente inteligente)

Tabla 3.1: Enfoques Inteligencia Artificial

Hoy en día, el enfoque dominante en la IA aplicada es el de los agentes racionales. Un agente racional es una entidad que percibe su entorno mediante sensores y actúa sobre él mediante actuadores, tomando decisiones que maximizan una medida de éxito o rendimiento. Este paradigma es especialmente útil en entornos dinámicos, como en el caso de los vehículos autónomos, robots agrícolas o sistemas de análisis de imágenes satelitales como el que se propone en este proyecto.

La IA puede clasificarse en dos grandes categorías según su alcance:

- **IA débil**: se refiere a sistemas diseñados para realizar tareas concretas, como jugar al ajedrez, recomendar películas o segmentar imágenes. Es el tipo de IA que se utiliza en este proyecto.
- **IA fuerte**: hace referencia a sistemas con capacidades generales equivalentes a las humanas, capaces de razonar, tener conciencia y emociones. Esta idea aún se encuentra en el terreno de la especulación filosófica y científica.

La IA abarca múltiples disciplinas y técnicas, entre ellas:

- **Aprendizaje automático (*Machine Learning*)**: algoritmos que aprenden patrones a partir de datos sin ser explícitamente programados. Hablaremos más en profundidad en la siguiente sección.
- **Procesamiento del lenguaje natural (*Natural Language Processing*)**: análisis y generación de lenguaje humano por parte de máquinas.
- **Visión artificial (*Computer Vision*)**: interpretación de contenido visual como imágenes o vídeo.
- **Robótica**: agentes físicos que interactúan con el mundo real.
- **Sistemas expertos**: sistemas que razonan con conocimiento codificado por expertos humanos.

Este proyecto se enmarca principalmente en visión por computador y aprendizaje profundo, ramas dentro del aprendizaje automático que permiten a un sistema "ver" y aprender a partir de imágenes.

La inteligencia artificial puede aplicarse en prácticamente cualquier ámbito en el que los seres humanos desempeñen tareas que impliquen cierto grado de inteligencia o toma de decisiones, siempre que dichas tareas puedan ser formalizadas y abordadas mediante algoritmos especializados. Aunque en la mayoría de los casos actuales la IA no ha alcanzado el nivel necesario para sustituir por completo a los profesionales humanos, sí representa una herramienta de gran valor que complementa y potencia el trabajo humano, mejorando la eficiencia, reduciendo errores y liberando tiempo para actividades de mayor nivel cognitivo.

Algunos ejemplos son:

- **Agricultura**: análisis de cultivos, predicción de cosechas, detección de plagas o segmentación de imágenes satelitales.
- **Medicina**: diagnóstico asistido por imagen, predicción de enfermedades o análisis genético. Los sistemas de IA pueden analizar grandes volúmenes de datos médicos y detectar patrones que pueden no ser evidentes para los médicos humanos, como en la tarea de realizar diagnósticos a partir de imágenes médicas.
- **Transporte**: vehículos autónomos, optimización de rutas.
- **Industria y robótica**: automatización de procesos, mantenimiento predictivo.
- **Finanzas**: detección de fraudes, predicción de mercados. Estos sistemas pueden analizar datos financieros en tiempo real y tomar decisiones de inversión más rápidamente que los humanos.
- **Educación, marketing, logística, videojuegos, ...**

3.3.2. Machine Learning

El *Machine Learning* (o aprendizaje automático) es uno de los pilares fundamentales de la inteligencia artificial moderna y constituye el núcleo de muchas aplicaciones actuales. Mientras que la inteligencia artificial busca simular capacidades humanas en general, el *Machine Learning* se centra en dotar a los sistemas de la capacidad de aprender de los datos y mejorar su desempeño a partir de la experiencia. En lugar de programar explícitamente cada paso que debe seguir un sistema para resolver una tarea, el enfoque del aprendizaje automático consiste en entrenar un modelo con ejemplos reales, para que este descubra patrones y pueda generalizar a nuevos casos. Este cambio de paradigma ha permitido abordar problemas complejos que son difíciles de formalizar mediante reglas, como el reconocimiento de imágenes, el análisis de voz o, como en este caso, la segmentación de cultivos en imágenes satelitales.

El aprendizaje automático puede definirse como:

- *"El proceso mediante el cual un sistema mejora su rendimiento en una tarea específica a partir de la experiencia."*
- *"Una rama de la inteligencia artificial que se ocupa del diseño y desarrollo de algoritmos que permiten a las máquinas mejorar su desempeño en una tarea, basándose en datos históricos."*

Este enfoque surge como respuesta a las limitaciones de la IA simbólica, basada en reglas lógicas codificadas manualmente. Aunque efectiva en entornos controlados, esta aproximación resultó ineficaz para problemas que los humanos resuelven de forma intuitiva (como el reconocimiento de rostros, el lenguaje natural o la clasificación de imágenes) pero que son extremadamente difíciles de formalizar en reglas.

En programación tradicional, un sistema recibe como entrada los datos y un programa (conjunto de reglas explícitas), y genera una salida. En *Machine Learning*, se invierte este enfoque: a partir de datos de entrada y salida conocidos, el sistema aprende automáticamente el programa (modelo) que describe la relación entre ambos.

Paradigma Tradicional	**Paradigma *Machine Learning***
Datos + Reglas → Resultado	Datos + Resultados → Reglas (modelo)

Tabla 3.2: Paradigma Tradicional vs Paradigma *Machine Learning*

Este modelo es especialmente útil cuando:

- No se conocen las reglas que gobiernan el problema.
- Existen grandes volúmenes de datos disponibles.

- La solución necesita adaptarse y mejorar con el tiempo.

El aprendizaje automático puede clasificarse en varios tipos según el tipo de datos disponibles y la forma en la que el sistema interactúa con ellos:

1. **Aprendizaje supervisado** Es el tipo más común de ML. El sistema recibe un conjunto de datos etiquetados, es decir, pares de entrada-salida, y aprende una función que mapea las entradas a las salidas esperadas.

   - **Objetivo**: predecir la salida para nuevas entradas no vistas.
   - **Ejemplos**:
     - Clasificación de correos como *spam* o no *spam*.
     - Diagnóstico médico a partir de imágenes.
     - Segmentación de imágenes de satélite (como en este TFG).

   Dos tipos principales:

   - **Clasificación**: la salida es una categoría discreta (viñedo / no viñedo). El algoritmo aprende con un conjunto de entrenamiento los patrones comunes de cada grupo para conseguir clasificar los elementos nunca vistos.
   - **Regresión**: la salida es un valor continuo (por ejemplo, estimación de rendimiento). Es de utilidad para conocer la relación de dependencia o no entre los datos existentes.

2. **Aprendizaje no supervisado** No se proporcionan etiquetas. El sistema debe descubrir estructuras ocultas en los datos, como agrupaciones o patrones.

   - **Objetivo**: identificar relaciones entre los datos.
   - **Ejemplos**:
     - Agrupar clientes según comportamiento de compra.
     - Detección de anomalías en sensores.
     - Análisis de imágenes sin anotaciones.

   Dos tipos principales:

   - ***Clustering***: consiste en buscar agrupaciones entre los datos.
   - **Reducción de dimensionalidad**: consiste en reducir la dimensionalidad de los datos sin perder información. Se suele utilizar antes del aprendizaje supervisado para visualizar o preprocesar los datos.

3. **Aprendizaje por refuerzo** El sistema interactúa con un entorno y aprende mediante un mecanismo de recompensas y penalizaciones.

- **Objetivo**: aprender una política de acción que maximice la recompensa acumulada.
- **Ejemplos**:
  - Robótica autónoma.
  - Juegos (como *AlphaGo*).
  - Gestión de recursos.

También existen enfoques mixtos como:

- **Aprendizaje semi-supervisado**: mezcla de datos etiquetados y sin etiquetar.
- **Aprendizaje por transferencia (*Transfer Learning*)**: se reutiliza el conocimiento aprendido en una tarea para aplicarlo en otra similar, lo cual es muy útil cuando no se dispone de suficientes datos. Este enfoque ha sido usado en el proyecto, hablaremos más en detalle en el siguiente capítulo.

Algunos de los algoritmos más representativos son *K-Nearest Neighbors (KNN)*, Árboles de decisión (*ID3, CART, C4.5*), *Random Forest, Support Vector Machines (SVM)*, redes neuronales (Perceptrón, *MLP, CNN*, de las que hablaremos en la siguiente sección), *Naive Bayes* o *K-means*.

Un proceso típico de aprendizaje automático incluye varias etapas:

1. **Recopilación de datos**: conjunto de ejemplos representativos del problema.
2. **Preprocesamiento**: limpieza (corregir datos erróneos, filtrar algunos datos incorrectos del *dataset* y reducir los detalles innecesarios en los datos), normalización (expresar datos en las mismas unidades de medida, escala o rango), identificación de ruido (detectar errores aleatorios o varianzas en las variables medidas), ...
3. **División del conjunto de datos**: en entrenamiento, validación y prueba.
4. **Entrenamiento del modelo**: se ajustan los parámetros del algoritmo.
5. **Evaluación del rendimiento**: se mide la capacidad de generalización.
6. **Ajuste de hiperparámetros**: para mejorar la precisión.
7. **Implementación y puesta en producción**

Algunos de los problemas comunes en *Machine Learning* son:

- **Sobreajuste (*overfitting*)**: el modelo se ajusta demasiado a los datos de entrenamiento, perdiendo capacidad de generalización.

- **Subajuste (*underfitting*)**: el modelo es demasiado simple y no capta la complejidad del problema.
- **Distribuciones desbalanceadas**: cuando hay muchas más muestras de una clase que de otra.
- **Datos ruidosos** o con errores.

En el caso de este TFG, el aprendizaje automático se aplica al desarrollo de un modelo de clasificación pixel a pixel sobre imágenes satelitales para identificar automáticamente filas de viñedo, lo cual se inscribe dentro del campo del aprendizaje supervisado y más concretamente en el ámbito de la segmentación semántica de imágenes.

3.3.3. Redes Neuronales

Las redes neuronales artificiales (*ANN*) son uno de los pilares fundamentales del aprendizaje automático moderno, especialmente en tareas que requieren procesamiento visual. Inspiradas en el funcionamiento del cerebro humano, estas estructuras computacionales permiten aprender patrones complejos a partir de datos, lo que las convierte en herramientas ideales para problemas donde la relación entre entrada y salida no es trivial o directamente no puede formalizarse mediante reglas.

Las redes neuronales son la base de muchas de las aplicaciones más exitosas de la inteligencia artificial actual: reconocimiento de voz, visión por computador, diagnósticos médicos, robótica, generación de texto e imagen, etc. En este proyecto, se emplea una red neuronal convolucional del tipo *U-Net*, un modelo especializado en segmentación semántica de imágenes, que forma parte del campo del aprendizaje profundo, una evolución natural de las redes neuronales tradicionales.

El concepto de red neuronal artificial se inspira en el sistema nervioso biológico, compuesto por neuronas interconectadas que transmiten señales entre sí. En el cerebro humano, una neurona se activa (dispara) cuando la señal acumulada que recibe supera cierto umbral. Esta idea se modeló matemáticamente por primera vez en 1943 con el modelo de McCulloch y Pitts, que representaba neuronas como nodos lógicos con entradas binarias, pesos fijos y una función de activación tipo escalón.

Aunque rudimentario, este modelo permitió simular funciones booleanas básicas (AND, OR, etc.) y sentó las bases para la construcción de redes artificiales más complejas.

Una neurona artificial recibe un conjunto de entradas numéricas $x_1, x_2, \ldots, x_n$, cada una asociada a un peso sináptico $w_1, w_2, \ldots, w_n$, que representan la importancia relativa de cada entrada. Estas entradas se combinan mediante una suma ponderada:

$$z = \sum_{i=1}^{n} w_i x_i + b$$

donde b es un término de sesgo (bias). El resultado se pasa a través de una función de activación $f(z)$, que limita la amplitud de la salida de la neurona (simulando el disparo

de la neurona):

$$y = f(z)$$

Algunas funciones de activación comunes incluyen:

- **Escalón**: salida 0 o 1 según si se supera un umbral.

$$Escalon(z) = \begin{cases} 0 & \text{si } z < \theta, \\ 1 & \text{si } z \geq \theta, \end{cases}$$

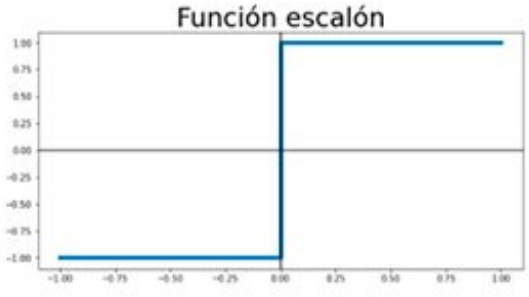


Figura 3.1: Función de Activación Escalón

- **Sigmoide**: salida entre 0 y 1, útil para clasificación binaria.

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

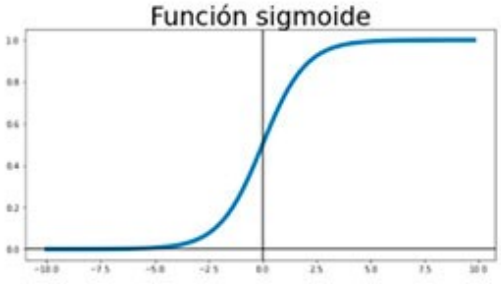


Figura 3.2: Función de Activación Sigmoide

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

$$ReLU(z) = \begin{cases} 0 & \text{si } z < 0, \\ z & \text{si } z \geq 0, \end{cases}$$

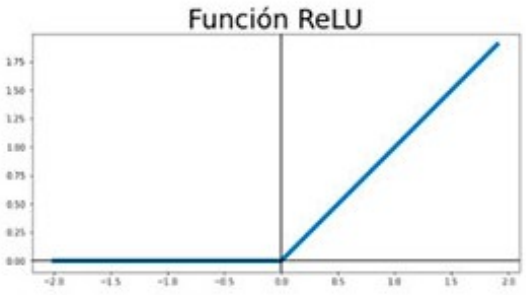


Figura 3.3: Función de Activación ReLU

- ***Softmax***: generalización para clasificación multiclase.

$$Softmax(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

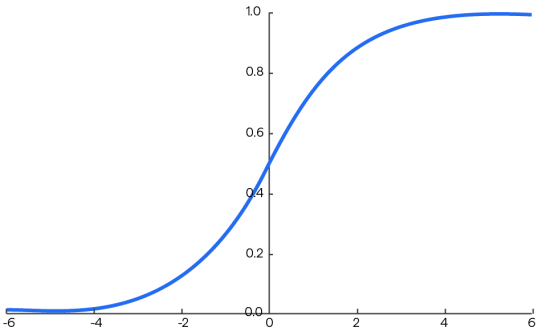


Figura 3.4: Función de *Softmax*

El perceptrón, propuesto por Frank Rosenblatt en 1958, fue uno de los primeros modelos de neuronas entrenables. Introdujo un mecanismo de ajuste automático de pesos mediante retroalimentación, basado en el error cometido al predecir la salida. Este ajuste se realiza siguiendo una fórmula como:

$$w_i \leftarrow w_i + \alpha(y_{real} - y_{predicho})x_i$$

donde α es la tasa de aprendizaje.

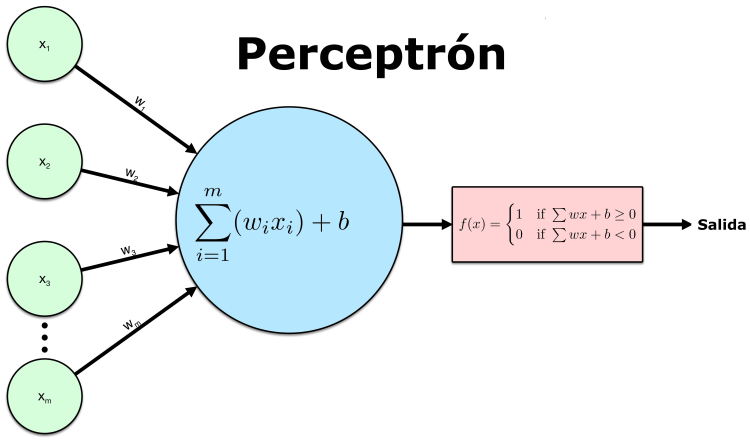


Figura 3.5: Modelo de Perceptrón

El perceptrón puede aprender a clasificar correctamente problemas linealmente separables, pero no puede resolver tareas como el XOR, lo que llevó a una pérdida de interés en el campo durante un tiempo.

Para superar las limitaciones del perceptrón simple, se introdujeron las redes neuronales multicapa (*Multilayer Perceptrons, MLP*). Estas redes están compuestas por:

- **Capa de entrada**: recibe los datos de entrada.
- **Capas ocultas**: realizan transformaciones intermedias.
- **Capa de salida**: proporciona el resultado final.

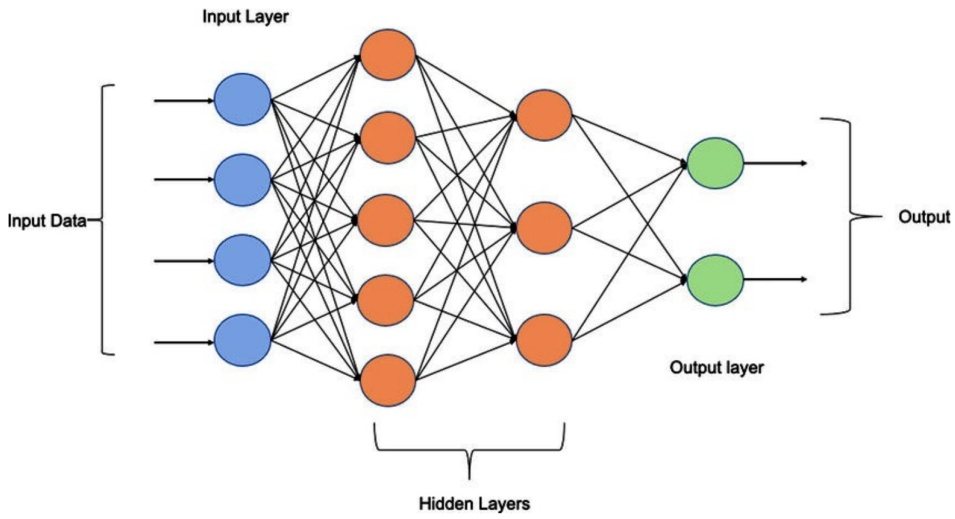


Figura 3.6: Red Perceptrón Multicapa (MLP)

Cada capa está completamente conectada con la siguiente y cada neurona opera como se describió antes. Al aumentar el número de capas y neuronas, la red gana capacidad de representación, siendo capaz de aprender funciones más complejas y no lineales.

En redes complejas determinar los pesos correctos de forma "manual" es imposible por eso los pesos y el bias son determinados mediante un proceso de aprendizaje que es lo que conocemos como entrenar la red.

El entrenamiento de una red neuronal multicapa se basa en la retropropagación del error (*backpropagation*), un algoritmo propuesto por Rumelhart, Hinton y Williams en 1986. Este proceso incluye:

1. **Propagación hacia adelante**: los datos pasan a través de la red para obtener una predicción.

2. **Cálculo del error**: se compara la predicción con la etiqueta real (por ejemplo, con una función de pérdida como el error cuadrático medio).

3. **Retropropagación**: se calculan las derivadas del error respecto a cada peso usando la regla de la cadena.

4. **Actualización de pesos**: se aplican pequeños ajustes para minimizar el error, normalmente usando descenso por gradiente.

Este proceso se repite durante múltiples épocas, y cada pasada por todos los datos permite que la red se ajuste progresivamente.

Existen múltiples arquitecturas, cada una adaptada a distintos tipos de tareas:

- **Redes Neuronales *Feedforward***
  - Usado para clasificación de datos estructurados.
  - Sus capas están completamente conectadas.
  - No tienen memoria ni estructura espacial.
- **Redes Neuronales Convolucionales (CNN)**
  - Específicas para procesamiento de imágenes.
  - Utilizan filtros locales que detectan bordes, texturas, formas.
  - Son la base de modelos como *U-Net* (usado en este TFG).

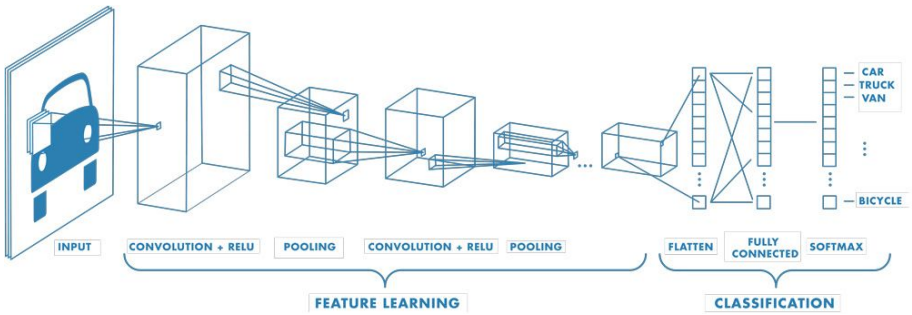


Figura 3.7: Red Neuronal Convolucional

- **Redes Neuronales Recurrentes (RNN)**

  - Modelan datos secuenciales (texto, voz, series temporales).
  - Permiten que sus salidas alimenten a sus entradas.
  - Incorporan memoria interna.
  - Variantes modernas: LSTM, GRU.

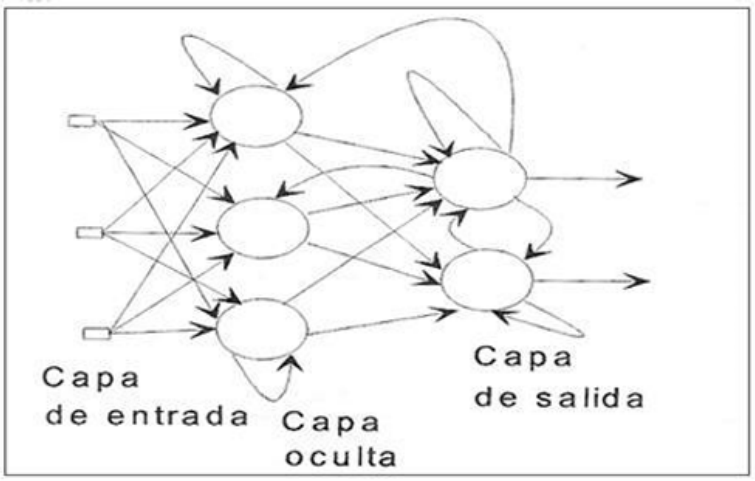


Figura 3.8: Red Neuronal Recurrente

- **Redes Autoorganizadas (SOM, mapas de Kohonen)**

  - Aprendizaje no supervisado.
  - Útiles para reducción de dimensionalidad y agrupamiento.

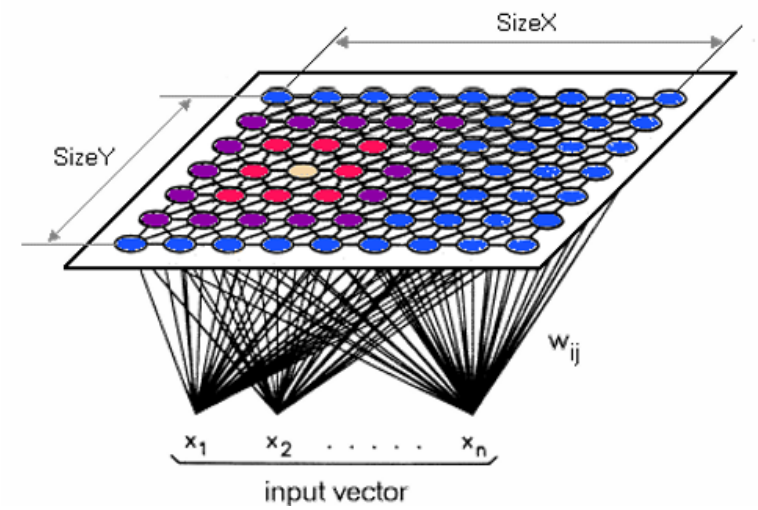


Figura 3.9: Red Autoorganizada

- **Redes Generativas (*GANs, autoencoders*)**
  - Aprenden representaciones latentes de los datos.
  - Usadas para generación de imágenes, mejora de resolución, etc.

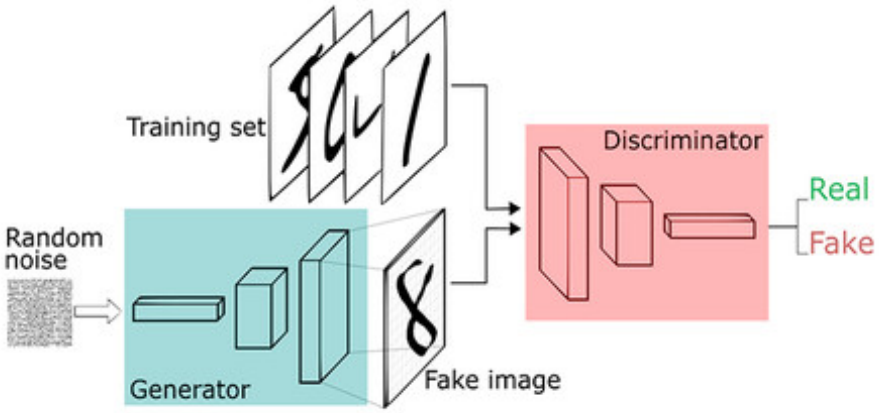


Figura 3.10: Red Generativa

Las redes neuronales tienes una serie de ventajas como son la capacidad de modelar relaciones no lineales complejas, la generalización, es decir, pueden predecir correctamente ejemplos no vistos o la robustez al ruido y a entradas incompletas.

Pero también tienen inconvenientes, como la necesidad de grandes volúmenes de datos para entrenar correctamente, la necesidad de grandes recursos computacionales, el riesgo de sobreajuste o la difícil interpretación de su funcionamiento interno (caja negra).

3.3.4. Deep Learning

El aprendizaje profundo, conocido como *Deep Learning (DL)*, es una subárea del aprendizaje automático (*Machine Learning*) que ha revolucionado la inteligencia artificial en la última década. Su capacidad para extraer patrones de alto nivel y realizar tareas complejas sin intervención humana directa lo ha convertido en el enfoque dominante para múltiples aplicaciones, incluyendo el procesamiento de imágenes, lenguaje natural, reconocimiento de voz y, en el contexto de este proyecto, la segmentación semántica de imágenes satelitales para análisis agronómico.

El *Deep Learning* se basa en los principios de las redes neuronales artificiales descritas previamente, pero con una diferencia clave: la profundidad del modelo. Mientras que las redes tradicionales suelen tener una o dos capas ocultas, las redes profundas pueden contar con decenas o incluso cientos de capas, capaces de representar funciones extremadamente complejas.

En una red profunda típica:

- Las primeras capas extraen características básicas (como bordes en una imagen).
- Las capas intermedias detectan combinaciones de características (formas, texturas).
- Las últimas capas capturan patrones abstractos (objetos completos, regiones semánticas).

Este procesamiento jerárquico es especialmente útil en visión por computador, donde los patrones visuales emergen de la combinación progresiva de elementos simples.

El desarrollo del *Deep Learning* ha seguido una evolución paralela al crecimiento computacional y al acceso a grandes volúmenes de datos:

- **Décadas de 1940–1980**: se sientan las bases con los modelos de McCulloch-Pitts, el perceptrón de Rosenblatt y los primeros estudios sobre redes neuronales multicapa.
- **Décadas de 1980–1990**: se introduce el algoritmo de retropropagación del error (*backpropagation*) y surgen las redes convolucionales (*CNN*) y las recurrentes (*RNN*), aunque sin gran impacto por limitaciones tecnológicas.
- **A partir de 2006**: se redescubren los modelos profundos gracias a:
  - El uso de GPU para acelerar el entrenamiento.
  - La disponibilidad de grandes *datasets* etiquetados.
  - Mejores algoritmos y técnicas de regularización.

- **2012**: la red *AlexNet* gana la competición *ImageNet*, marcando un punto de inflexión y consolidando el *Deep Learning* como tecnología líder.

Los componentes clave en una red profunda son los siguientes:

1. **Capas de procesamiento**
   - **Capas densas (*fully connected*)**: cada neurona está conectada a todas las neuronas de la capa anterior.
   - **Capas convolucionales (*CNNs*)**: usan filtros locales para detectar patrones espaciales.
   - **Capas recurrentes (*RNNs*)**: incorporan memoria para datos secuenciales.
2. **Funciones de activación**
   Permiten introducir no linealidades. Las más usadas en *Deep Learning* son:
   - **ReLU**: de la que ya hemos hablado en la sección anterior, rápida y efectiva, evita el *vanishing gradient* (para valores positivos).
   - **Sigmoide y tanh**: menos utilizadas actualmente por saturación.
   - ***Softmax***: convierte los valores de salida en probabilidades (valores numéricos en el rango [0,1] que son más fácil de interpretar). La salida será la probabilidad de ocurrencia de cada una de las salidas posibles para los modelos de clasificación.
3. **Función de pérdida (*Loss Function*)**
   Evalúa el error entre la predicción y la salida real. Es la versión más avanzada de lo que ya hacíamos con el perceptrón. El entrenamiento se realiza minimizando progresivamente esta función de pérdida, por lo que si la red no clasifica muy bien el conjunto de datos, la *loss function* tendrá un valor alto:
   - ***Cross-entropy***: usada en tareas de clasificación. Mide la distancia entre dos distribuciones de probabilidad por lo que sólo se puede aplicar sobre distribuciones de probabilidad (por ejemplo, después de una capa de salida del tipo *Softmax*).
   - ***MSE (Mean Squared Error)***: para regresión.
4. **Optimizadores**
   Algoritmos que ajustan los pesos para minimizar la *loss function*. El algoritmo básico de optimización en redes neuronales es del *gradient descent* (descenso por gradiente). Existen multitud de algoritmos de optimización que podemos utilizar y configurar a la hora de crear nuestras redes:
   - ***SGD (Stochastic Gradient Descent)***
   - ***Adam, Adagrad, RMSProp, ...***

5. **Regularización**
   Evita el sobreajuste en redes muy grandes realizando pequeñas modificaciones sobre el algoritmo de aprendizaje de tal manera que este generaliza mejor. Los principales tipos de regularización son:

   - ***Dropout***: desactiva aleatoriamente neuronas durante el entrenamiento.

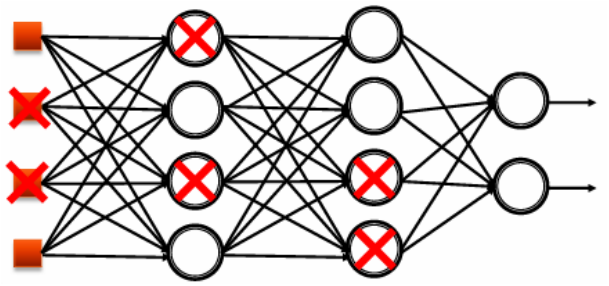


Figura 3.11: Ejemplo de *Dropout*

   - ***Early Stopping***: detiene el entrenamiento si no mejora el rendimiento en validación tras un número determinado de épocas.
   - ***L2 & L1 regularization***: es el método básico de regularización. Añaden un término de regularización a la *loss function*, lo que hace que los pesos de las matrices disminuyan más de lo que deben, teniendo así modelos más sencillos y por lo tanto con menor riesgo de sobreajuste.
   - ***Data Augmentation***: amplía artificialmente el *dataset* (rotación, escalado, ...). Cuanto mayor sea el conjunto de entrenamiento menores posibilidades hay de sobreajuste.

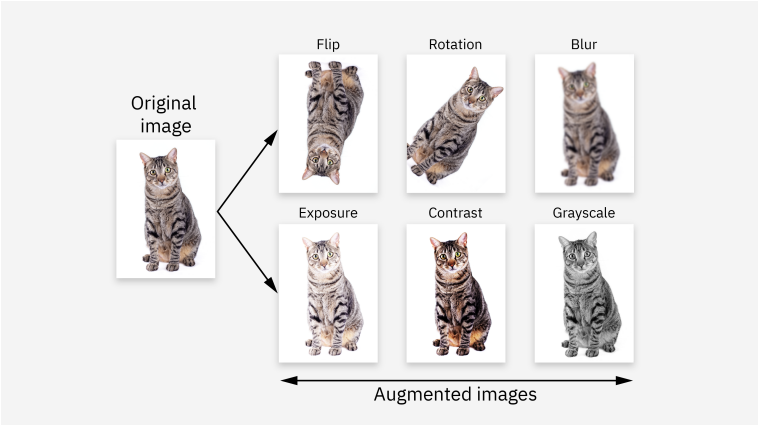


Figura 3.12: Ejemplo de *Data Augmentation*

3.3.5. Redes Neuronales Convolucionales

Son el tipo de red más utilizado en *Deep Learning* para tareas de visión por computador. Se basan en la idea de que una imagen contiene estructuras locales repetitivas que pueden ser detectadas por filtros. Una imagen se puede representar como una matriz de los valores de los pixels que forman la imagen, valores entre 0 y 255. Además, si la imagen es en color, tiene tres canales (*red, green and blue*) que le da el volumen pero si la imagen es en escala de grises tendrá un único canal.

Una *CNN* básica tiene la siguiente forma:

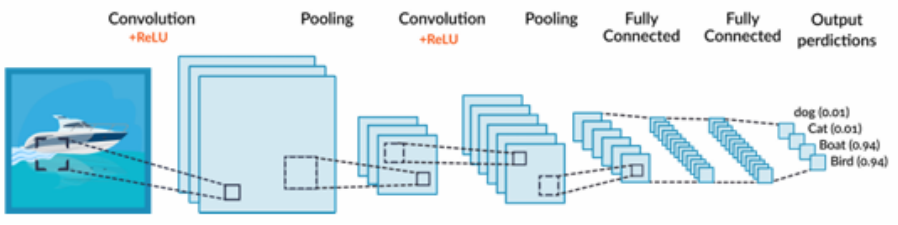


Figura 3.13: Arquitectura básica de una *CNN*

Es decir, sus componentes principales son:

- **Capas de convolución**: la operación de convolución es la encargada de extraer las características de la imagen. Un filtro o *kernel* (habitualmente de tamaño 3x3 o 5x5) se pasa sobre la imagen. La operación de convolución multiplica los pixels de la imagen por los pesos definidos en el filtro y se suma el resultado total. Existen tres parámetros fundamentales que podemos configurar a la hora de aplicar la convolución:

  - ***Depth***: número de filtros que aplicamos (suele ser habitual utilizar 32 filtros, 64 filtros, ...). Cuantos más filtros utilicemos, más mapas de características generaremos.
  - ***Stride***: número de pixels que saltamos al movernos. Un mayor valor de *stride* produce mapas de características más pequeños.
  - ***Padding***: consiste en añadir un marco de ceros alrededor de la imagen para poder aplicar mejor los filtros sobre los elementos de los bordes.

  Distintos filtros pueden tener distintos efectos en las imágenes, generando así distintos mapas de características.

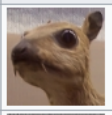
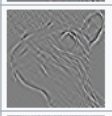
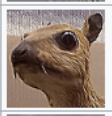
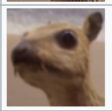
Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge or edge detection	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Figura 3.14: Ejemplos de diferentes *Kernels* aplicados a la misma imagen

Después aplicamos la función de activación ReLU a la matriz para eliminar los valores negativos y dejar los positivos como están.

- **Capas de *Pooling***: estas capas se utilizan para reducir el tamaño de la matriz. Se aplica un filtro sobre la salida de la capa anterior y selecciona un único número como salida. El objetivo de esto es que la red sea capaz de entrenar más rápido. Existen distintos tipos de *pooling*:

  - ***Max Pooling***: selecciona el valor máximo. Suele ser el que mejor funciona y se suele aplicar con un filtro de 2x2 y *stride* de 2.
  - ***Average Pooling***: selección el valor medio.
  - ***Sum Pooling***: selecciona la suma de los valores.

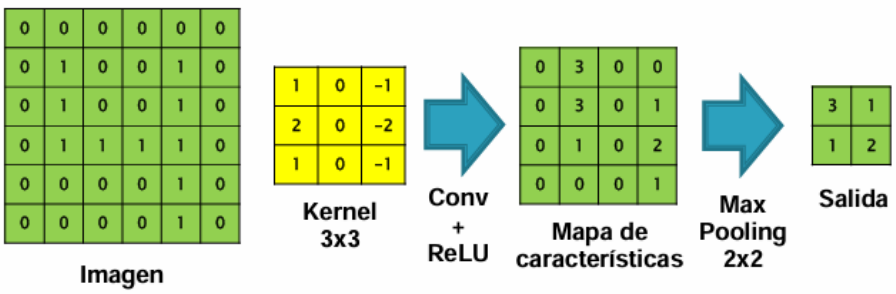


Figura 3.15: Ejemplo de aplicación de una capa formada por Convolución + ReLU + *Max Pooling*

- **Capa *Fully Connected***: es una capa neuronal multicapa normal que se utiliza para transformar la matriz tridimensional de características que se genera en las convoluciones en un vector unidimensional. El objetivo es combinar de manera eficiente todas las características extraídas en las capas anteriores.

  A ese vector unidimensional se le aplica posteriormente *Softmax* para que nos de la probabilidad de que la salida sea de una clase o de otra.

Existen varias arquitecturas de CNNs "precreadas", arquitecturas que han funcionado de manera exitosa en problemas y concursos de clasificación de imágenes (como el de *Imagenet*) y que ahora podemos importar y utilizar. En la mayoría de los casos, son redes que ya vienen preentrenadas y que con un pequeño entrenamiento, pocas épocas y sin un número excesivo de imágenes podemos obtener unos resultados muy competitivos.

Algunas de estas arquitecturas son *AlexNet* (2012), la cual ya hemos mencionado anteriormente, que tiene un total de aproximadamente 60 millones de parámetros a entrenar y fue una de las primeras CNNs entrenadas en GPU. *VGG16* y *VGG19* (2014), que fueron desarrolladas para ganar a *AlexNet* en la competición de *ImageNet*, contando con 138 millones de parámetros. *ResNet* (2015), simple y eficiente, de lo mejor que podemos usar (sus pesos ha sido usados para el entrenamiento del modelo que se desarrolla en este proyecto mediante la técnica *Transfer Learning*). Hay versiones con 34, 50, . . . , hasta 152 capas. O *U-Net* (2015), que ha sido la arquitectura utilizada para el entrenamiento del modelo que aquí se desarrolla, especializada en segmentación semántica. Hablaremos más profundamente de ella en el siguiente capítulo.

3.3.6. Segmentación

La segmentación de imágenes es una técnica fundamental en el campo de la visión por computador y una de las tareas más relevantes dentro del aprendizaje profundo aplicado al análisis visual. Su objetivo principal es dividir una imagen en regiones coherentes que representen objetos, estructuras u otras entidades significativas. A diferencia de la clasificación tradicional, que asigna una sola etiqueta a toda una imagen, la segmentación trabaja a nivel de píxel, proporcionando una descripción mucho más detallada y localizada del contenido visual.

Antes del auge del *Deep Learning*, la segmentación se realizaba mediante métodos clásicos de procesamiento de imágenes:

- **Umbralización (*thresholding*)**: binariza la imagen según un valor umbral (global o adaptativo).
- **Detección de bordes**: detecta contornos mediante operadores como *Sobel*, *Canny* o *Laplaciano*.
- **Regiones crecientes (*region growing*)**: agrupa píxeles similares a partir de semillas.

- ***Clustering***: algoritmos como *K-means* para agrupar píxeles por color o textura
- **Modelos activos (*snakes*)**: contornos que se ajustan dinámicamente a los bordes.

Estos métodos son rápidos, pero suelen ser sensibles al ruido, la iluminación o las variaciones en la escena. Además, requieren una fuerte intervención manual y no se adaptan bien a imágenes complejas como las satelitales multiespectrales.

Con el desarrollo del *Deep Learning*, se han propuesto arquitecturas especializadas en segmentación que superan con creces las limitaciones de los métodos tradicionales. Estas redes, típicamente convolucionales, pueden aprender directamente de los datos cómo segmentar las distintas clases presentes en una imagen, con alta precisión.

Las arquitecturas más destacadas incluyen:

- ***U-Net***: red en forma de U, con codificador-decodificador y *skip connections*. Ideal para segmentación precisa en contextos con pocos datos.
- ***SegNet***: similar a *U-Net*, con codificación-decodificación y uso de índices de *pooling*.
- ***DeepLab (v3, v3+)***: incorpora convoluciones dilatadas y módulos de atención espacial.
- ***Mask R-CNN***: permite segmentación por instancias.

Existen diferentes enfoques y niveles de segmentación, dependiendo del nivel de detalle y del tipo de información que se desea extraer:

- **Segmentación semántica**
  Consiste en asignar a cada píxel de la imagen una etiqueta correspondiente a una clase. No se diferencian instancias individuales, solo se identifica qué regiones pertenecen a qué clase.
  - Ejemplo: todos los píxeles correspondientes a "viñedo" tienen la misma etiqueta.
  - Este es el enfoque utilizado en este proyecto.

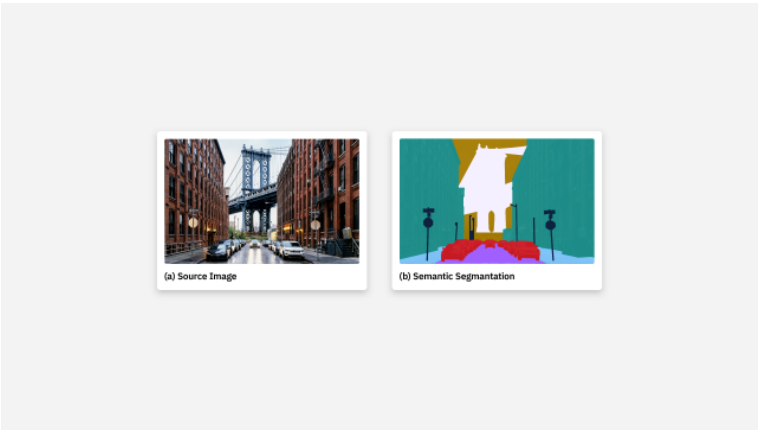


Figura 3.16: Ejemplo de Segmentación Semántica

- **Segmentación de Instancias**
  Va un paso más allá de la segmentación semántica: no solo clasifica los píxeles, sino que también distingue entre distintas instancias del mismo objeto.
  - Ejemplo: identificar y separar cada fila de viñedo de forma individual.
  - Requiere modelos más complejos (*Mask R-CNN*, ...).

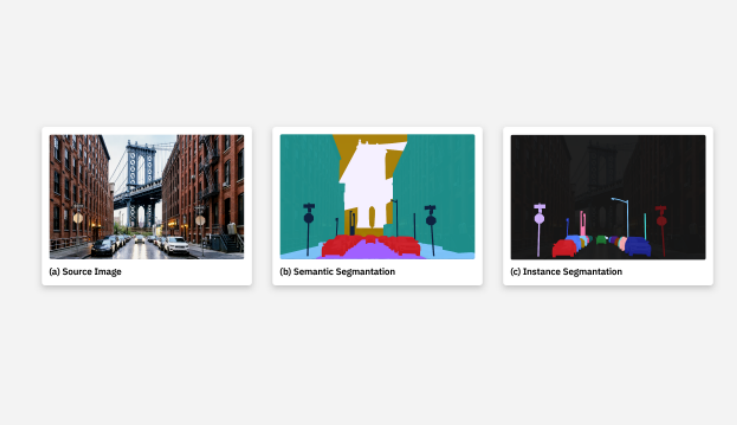


Figura 3.17: Ejemplo de Segmentación de Instancias

- **Segmentación Panóptica**
  Combina los dos tipos anteriores: clasifica todos los píxeles (como la semántica) y además identifica instancias únicas (como la de instancias). Es una solución completa, pero computacionalmente costosa.

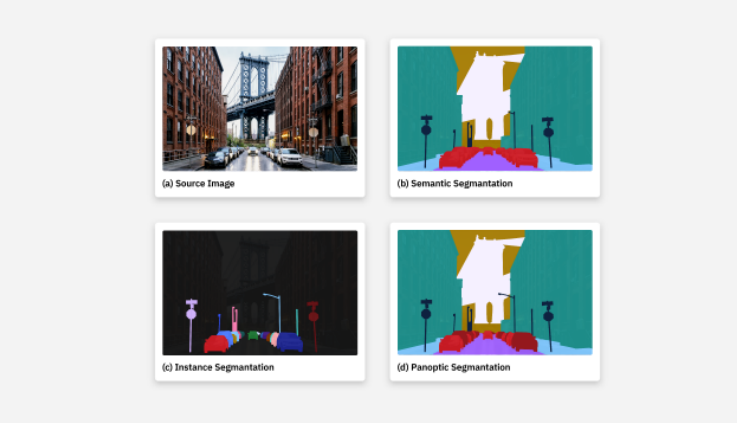


Figura 3.18: Ejemplo de Segmentación Panóptica

Para medir el rendimiento de un sistema de segmentación, se utilizan métricas específicas que comparan las máscaras predichas con las máscaras reales (etiquetadas manualmente). Las más comunes son:

- **Precisión (*Accuracy*)**: proporción de píxeles correctamente clasificados.
- ***IoU (Intersection over Union)***: mide la superposición entre entre el polígono delimitador predicho por el modelo y el polígono delimitador real del objeto en la imagen. Si llamamos A la predicción y B a la anotación real:

$$IoU = \frac{A \cap B}{A \cup B}$$

  De la definición podemos ver que $0 \leq IoU \leq 1$, por lo tanto, cuanto más se acerque a 1, la predicción es más certera.
- ***Dice Coefficient (F1 Score)***: otra medida de superposición, más sensible a clases minoritarias.

3.4. Fundamentos técnicos

Además de comprender los fundamentos teóricos que sustentan la solución propuesta, resulta indispensable describir y justificar los elementos técnicos que permiten su implementación efectiva. Esta sección recoge los aspectos clave del entorno tecnológico empleado para el desarrollo del sistema de segmentación automática de filas de viñedo a partir de imágenes satelitales.

Se detallan en primer lugar los lenguajes de programación utilizados, así como las librerías y herramientas software que han facilitado tanto el preprocesamiento de datos como la construcción, entrenamiento y evaluación del modelo de inteligencia artificial. A continuación, se analizan los *datasets* empleados para entrenar y validar el sistema, prestando especial atención a su procedencia, estructura y adecuación al problema planteado.

El propósito de esta sección es proporcionar una visión completa y justificada del ecosistema técnico sobre el que se ha construido el sistema, asegurando la reproducibilidad, escalabilidad y viabilidad práctica del desarrollo realizado.

3.4.1. Lenguajes y librerías

El desarrollo de soluciones basadas en inteligencia artificial, y más concretamente en aprendizaje profundo, exige el uso de lenguajes de programación que ofrezcan flexibilidad, eficiencia y un ecosistema maduro de herramientas científicas. Aunque existen múltiples lenguajes utilizados en la práctica, algunos destacan especialmente en el ámbito del *Machine Learning* y la visión por computador:

- **Python**: es el lenguaje más extendido y popular en la comunidad de ciencia de datos e IA. Su sintaxis clara, la enorme disponibilidad de librerías especializadas, y su integración con plataformas como *Jupyter Notebooks o Google Colab* lo convierten en la opción preferida para prototipado, investigación y despliegue.

- **R**: muy utilizado en estadística y análisis de datos, aunque con menor presencia en visión por computador o redes neuronales profundas.
- **Julia**: emergente en el campo del cómputo científico, con gran rendimiento, pero aún con una comunidad más reducida.
- **C++/CUDA**: usado en desarrollo de bajo nivel y optimización de librerías, especialmente para aprovechar la potencia de GPUs.

El presente proyecto ha sido desarrollado íntegramente en Python, aprovechando su ecosistema rico y robusto de librerías orientadas a tareas de aprendizaje profundo, procesamiento de imágenes y manipulación de datos. Python permite combinar de forma sencilla componentes de diferentes niveles, desde el preprocesamiento de datos hasta el diseño y entrenamiento de modelos complejos de redes neuronales como *U-Net*.

A continuación se describen las principales librerías empleadas en el proyecto, agrupadas por funcionalidad:

- **Librerías de modelado y entrenamiento de redes neuronales**
  Estas librerías permiten definir, entrenar y evaluar modelos de *deep learning* de forma modular, intuitiva y eficiente, facilitando el uso de GPU y aceleradores de *hardware*.
  - ***PyTorch (torch, torchvision)***: es uno de los *frameworks* más potentes y flexibles para *Deep Learning*. Fue desarrollado por *Facebook AI Research (FAIR)* y destaca por su facilidad para prototipar redes neuronales, su uso de tensores dinámicos y su integración con GPU. Es especialmente popular en entornos académicos y de investigación.
  - ***Torchvision***: es una extensión de *PyTorch* especializada en visión por computador. Incluye arquitecturas preentrenadas (como *ResNet*), transformaciones de imágenes, y herramientas para cargar *datasets* visuales estándar.
- **Librerías para procesamiento y análisis de imágenes**
  La manipulación de imágenes es un paso esencial en la mayoría de los proyectos de visión por computador, especialmente cuando se trabaja con datos satelitales o imágenes RGB.
  - ***OpenCV (cv2)***: biblioteca de visión por computador de código abierto ampliamente utilizada en tareas de preprocesamiento de imágenes, detección de contornos, transformaciones geométricas y procesamiento morfológico. Soporta operaciones en tiempo real y acceso a bajo nivel.
  - ***Pillow (PIL.Image)***: derivada del antiguo proyecto *PIL (Python Imaging Library)*, permite trabajar con imágenes en múltiples formatos y realizar tareas básicas como apertura, redimensionamiento, rotación o conversión entre modos de color.

- **Librerías para manipulación de datos numéricos y estructurados**
  Estas librerías permiten cargar, limpiar, transformar y analizar datos, tanto en forma de *arrays* numéricos como de estructuras tabulares. Son fundamentales en todo *pipeline* de *Machine Learning*.

  - ***NumPy***: núcleo del cálculo numérico en Python. Proporciona estructuras de datos eficientes como los *arrays* multidimensionales (tensores) y operaciones algebraicas vectorizadas, necesarias para trabajar con imágenes y pesos de redes neuronales.
  - ***Pandas***: herramienta de análisis de datos que permite manipular conjuntos de datos estructurados como tablas (*DataFrames*), facilitando la lectura de archivos *CSV/Excel* y el análisis exploratorio. Es especialmente útil para cargar anotaciones, gestionar rutas de imágenes y realizar estadísticas.

- **Librerías para entrenamiento, evaluación y seguimiento**

  - ***scikit-learn***: aunque centrada en algoritmos de *machine learning* clásico (árboles, SVM, *clustering* etc.), se ha utilizado en este proyecto para tareas complementarias como la división de datos o el cálculo de métricas.
  - ***tqdm***: facilita la monitorización de procesos largos (como el entrenamiento de modelos o carga de datos) mediante una barra de progreso interactiva en terminal o *notebook*.
  - ***matplotlib***: principal librería de visualización en Python. Permite representar gráficamente los resultados de entrenamiento (curvas de pérdida, precisión, etc.), imágenes originales y segmentadas, y comparativas entre predicción y verdad.

- **Librerías auxiliares y del sistema**

  - ***os, re, json***: librerías estándar de Python que permiten interactuar con el sistema de archivos, procesar nombres de archivos con expresiones regulares y leer o escribir anotaciones en formato JSON, respectivamente.

  Estas herramientas, aunque no están directamente relacionadas con el modelado de IA, son esenciales para construir *scripts* automatizados, control de rutas, gestión de datos de entrenamiento, entre otros aspectos prácticos del desarrollo.

Gracias al conjunto de librerías utilizadas, el desarrollo del sistema ha podido realizarse de forma eficiente, escalable y reproducible, manteniendo una clara separación entre lógica de negocio, modelado, procesamiento de datos y visualización. Esto no solo acelera el tiempo de desarrollo, sino que permite fácilmente adaptar el sistema a nuevas tareas, reutilizar componentes o integrarlo en flujos de trabajo mayores.

Además, muchas de estas librerías cuentan con documentación extensa, comunidades activas y compatibilidad con servicios en la nube (como *Google Colab*), lo que reduce drásticamente las barreras de entrada para estudiantes e investigadores que deseen trabajar en proyectos complejos como el presente TFG.

3.4.2. *Datasets* para segmentación

Uno de los aspectos más críticos en el desarrollo de sistemas de aprendizaje profundo es la disponibilidad de datos adecuados para entrenar y evaluar los modelos. En tareas de visión por computador, y más concretamente en segmentación semántica, se requiere no solo imágenes de alta calidad, sino también etiquetas a nivel de píxel, lo que implica un proceso de anotación laborioso y altamente especializado.

En este contexto, disponer de *datasets* representativos y correctamente anotados es fundamental para lograr modelos precisos, robustos y capaces de generalizar a nuevos escenarios. No obstante, en el ámbito agrícola (y más aún en aplicaciones específicas como la segmentación de filas de viñedos) la oferta de *datasets* públicos es muy limitada. Esta carencia ha condicionado directamente la estrategia seguida en este proyecto, obligando a la creación de un conjunto de datos propio, adaptado específicamente al problema planteado.

Un *dataset* orientado a tareas de segmentación semántica consta, habitualmente, de dos componentes:

- **Imágenes** (en formato RGB o multiespectral), que representan las escenas a analizar.
- **Máscaras de segmentación** (etiquetadas), en las que cada píxel tiene asignada una clase.

En tareas binarizadas, como la del presente TFG, las máscaras tienen dos valores posibles:

- $1 \rightarrow$ píxel correspondiente a "fila de viñedo".
- $0 \rightarrow$ píxel correspondiente a "fondo" o "no viñedo".

Para que el aprendizaje sea eficaz, es deseable que el *dataset* presente:

- Variedad geográfica (diferentes zonas vitícolas).
- Variabilidad temporal (distintas épocas del año).
- Diversidad visual (tipos de suelo, orientación, densidad, resolución).

En el caso de cultivos como el viñedo, esta diversidad es clave, ya que su aspecto varía significativamente en función del estado fenológico, la densidad de plantación o la orientación de las hileras.

Durante la fase inicial del proyecto, se realizó una revisión de *datasets* públicos orientados a segmentación de cultivos, como:

- ***Sentinel-2 crops (copernicus)*** [45]: imágenes multiespectrales orientadas a clasificación de parcelas, pero con resolución insuficiente para detectar filas individuales.
- ***Treibacher dataset o Drones for Agriculture***: centrados en otros tipos de cultivos (trigo, maíz, girasol) o en imágenes de drones, pero sin segmentación específica de hileras.
- ***VinePhen dataset***: orientado a análisis fenotípico de vides, pero con imágenes de detalle (no satelitales) y sin etiquetas completas para segmentación semántica.
- ***DeepWeeds, CropDeep*** [16]: para detección de malas hierbas o reconocimiento de especies, sin información estructural sobre la organización del cultivo.

En todos los casos analizados, se constató que ninguno de estos conjuntos de datos ofrecía imágenes satelitales con máscaras de segmentación centradas en la disposición estructural del viñedo, es decir, no eran útiles directamente para entrenar un modelo que identificara la disposición lineal de las hileras en una parcela agrícola.

Ante la ausencia de *datasets* adecuados, se optó por construir un conjunto de datos personalizado, adaptado específicamente al problema de la segmentación de filas de viñedos en imágenes aéreas. Para ello, se recurrió a fuentes accesibles y gratuitas de imágenes de alta resolución:

- **Sede Electrónica del Catastro (Dirección General del Catastro, España)**
  Proporciona ortofotos actualizadas y georreferenciadas de gran parte del territorio español, con suficiente resolución para identificar visualmente las hileras de viñedo.
- ***Google Maps / Google Earth***
  Se emplearon capturas de pantalla geolocalizadas para completar zonas específicas y obtener variedad visual en cuanto a tipos de cultivo, iluminación o geometría.

Las imágenes fueron descargadas, recortadas y preprocesadas manualmente para obtener fragmentos representativos de parcelas vitivinícolas con disposición en hileras bien definida.

Dado que no existen etiquetas oficiales para este tipo de segmentación, se realizó un proceso manual de etiquetado pixel a pixel:

- Se generaron máscaras binarias en las que se dibujaron las filas de viñedo visibles en cada imagen.
- El etiquetado se hizo mediante herramientas como *CVAT (Computer Vision Annotation Tool)* y edición directa de máscaras en formato PNG, asegurando la alineación con las imágenes originales.
- La validación de este etiquetado se realizó de forma visual, contrastando en el terreno que el número y disposición de las hileras coincidían con lo etiquetado en las imágenes.

Algunas de las ventajas de tener un *dataset* personalizado son:

- **Alta especificidad**: orientado exactamente a la tarea objetivo (filas de viñedos).
- **Control sobre el contenido**: posibilidad de ajustar calidad, resolución y equilibrio de clases.
- **Escalabilidad**: posibilidad de ampliar el *dataset* progresivamente con nuevas capturas.
- **Adaptabilidad**: se puede extender a otros cultivos en hileras mediante cambios mínimos.

Pero también existen inconvenientes:

- **Tiempo de etiquetado**: crear máscaras precisas manualmente es un proceso muy lento.
- **Variedad limitada**: al ser un *dataset* reducido y centrado en ciertas regiones, podría haber sesgo geográfico o visual.
- **Problemas de iluminación y resolución**: algunas imágenes satelitales presentan sombras, nubes o texturas complejas que dificultan el etiquetado.

Parte II

Desarrollo de la solución y resultados

Capítulo 4

Desarrollo de la solución y experimentación

Este capítulo describe en detalle el proceso de construcción del sistema de segmentación automática de filas de viñedos, desde el planteamiento técnico inicial hasta su implementación y puesta en funcionamiento. Se aborda cómo se ha materializado la solución planteada, con especial énfasis en las decisiones de diseño, la arquitectura del modelo de aprendizaje profundo implementado, la configuración experimental y el proceso de entrenamiento.

El desarrollo de esta solución ha requerido una fase de experimentación iterativa en la que se ha probado, ajustado y evaluado la eficacia de distintos componentes del sistema. A través de pruebas sistemáticas se ha determinado la configuración óptima de hiperparámetros, la estructura de red más adecuada, así como la estrategia de entrenamiento y validación.

Además, se explican los procedimientos seguidos para preparar los datos de entrada, generar las máscaras de segmentación y visualizar los resultados del modelo entrenado, mostrando cómo se ha ido refinando el sistema hasta obtener un comportamiento satisfactorio en términos de precisión y generalización.

4.1. Diseño experimental

El diseño experimental constituye la base sobre la que se sustenta el desarrollo y validación del sistema de inteligencia artificial. En esta sección se definen los elementos fundamentales que estructuran la experimentación, incluyendo el conjunto de datos utilizado, los criterios seguidos para su preparación y partición, y la selección inicial de hiperparámetros del modelo.

Este diseño no solo establece las condiciones necesarias para entrenar y evaluar el modelo de segmentación, sino que también garantiza que los resultados obtenidos sean reproducibles, comparables y significativos desde el punto de vista técnico. A través de una planificación cuidadosa del proceso experimental se ha buscado maximizar la eficiencia

del entrenamiento y asegurar la representatividad de las muestras, minimizando el riesgo de sobreajuste y permitiendo una evaluación objetiva del rendimiento del sistema.

4.1.1. Conjunto de datos

Uno de los principales retos en este Trabajo de Fin de Grado ha sido la ausencia de *datasets* públicos adecuados para el problema específico abordado. Si bien existen numerosos conjuntos de datos centrados en clasificación de cultivos, segmentación de parcelas agrícolas o análisis fenotípico, la gran mayoría no proporcionan máscaras de segmentación específicas de hileras de viña ni están pensados para imágenes satelitales con suficiente resolución.

Por esta razón, se optó por construir un conjunto de datos propio, utilizando imágenes extraídas manualmente de dos fuentes públicas y accesibles:

- **Sede Electrónica del Catastro (España)**: ofrece ortofotos georreferenciadas de alta resolución, ideales para la identificación de estructuras agrarias.
- ***Google Maps / Google Earth***: complementaron el *dataset* con imágenes aéreas actuales de diferentes regiones vitivinícolas.

Este enfoque permite disponer de datos visualmente representativos, realistas y variados, con control total sobre su calidad, formato y anotación.

El conjunto de datos está formado por pares de imágenes RGB y sus correspondientes máscaras binarias, donde cada píxel está etiquetado como:

- **1**: píxel perteneciente a una fila de viñedo.
- **0**: píxel correspondiente a fondo o zona no cultivada.

Las características principales del *dataset* son:

Atributo	**Valor aproximado**
Nº Total de Imágenes	150
Resolución	1280 × 1280 px
Formato	PNG
Tipo de imagen	RGB
Tipo de máscara	Binaria (1 canal)
Canales de entrada	3 (R, G, B)

Tabla 4.1: Características principales del *dataset*

Cada imagen ha sido preprocesada para uniformizar la resolución, normalizar los colores y garantizar la alineación exacta con su máscara, lo cual es fundamental para el entrenamiento supervisado del modelo.

Dado que no existía un *dataset* anotado previamente, fue necesario realizar la segmentación manual de las filas de viñedo. Para ello se utilizaron herramientas especializadas en anotación como *CVAT (Computer Vision Annotation Tool)*, una herramienta online que permite crear máscaras pixel a pixel mediante dibujo sobre la imagen.

Esta herramienta permite dibujar polígonos sobre las imágenes para delimitar el contorno de las filas de viñedos. Dado el elevado número de filas presentes en cada imagen, se optó por una solución práctica: anotar cada fila mediante un polígono simple de cuatro vértices. Si bien esta forma no representa con total precisión la geometría real de las hileras, resultó ser un compromiso necesario para poder etiquetar todo el conjunto de datos en un tiempo razonable, evitando un proceso extremadamente laborioso de anotación pixel a pixel.

Un ejemplo de anotación es el siguiente:

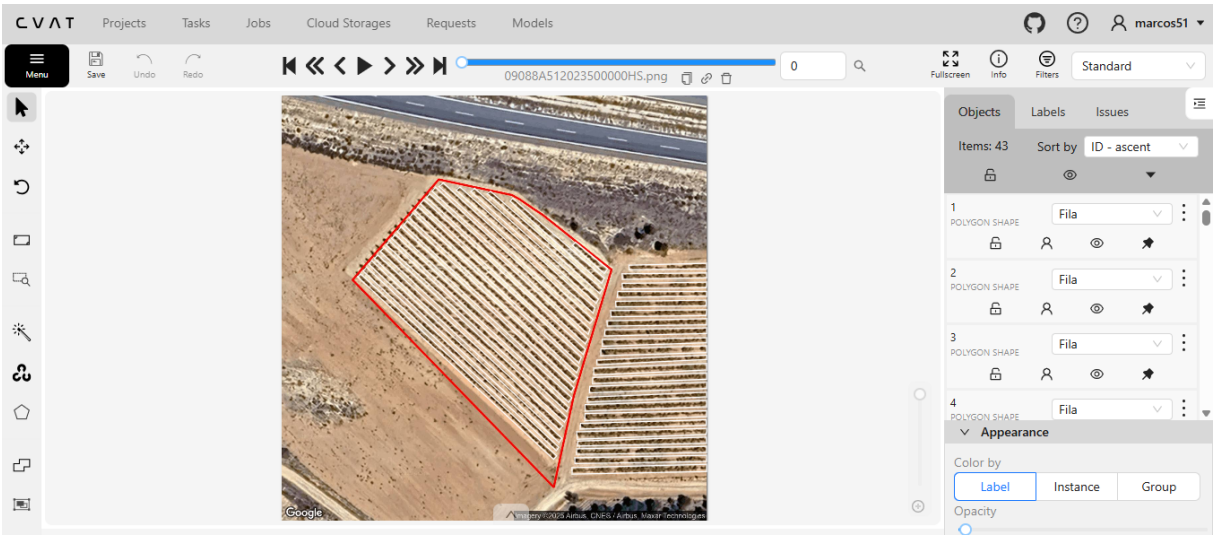


Figura 4.1: Ejemplo de anotación manual con la herramienta CVAT

La calidad de la anotación se realizó de forma visual, contrastando en el terreno que el número y disposición de las hileras coincidían con lo etiquetado en las imágenes.

Previamente, se exploraron herramientas de segmentación automática con la esperanza de acelerar el proceso, pero ninguna de las soluciones evaluadas ofrecía resultados con la calidad o precisión necesarias para este proyecto.

Las máscaras se generaron como imágenes en escala de grises, que posteriormente fueron binarizadas (umbral de 127) y convertidas a formato de un solo canal para su uso en el modelo.

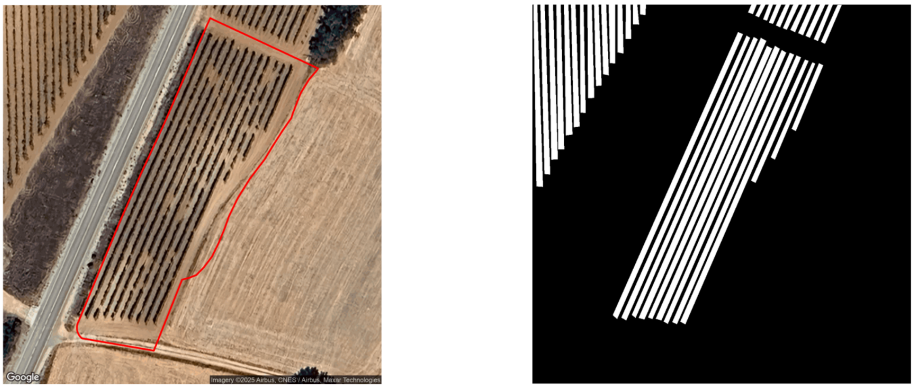


Figura 4.2: Ejemplo imagen original vs máscara anotada manualmente

Además, en el caso de las imágenes destinadas al conjunto de test (imágenes que no fueron utilizadas para entrenar), se aplicó un preprocesamiento adicional con el objetivo de eliminar visualmente las filas de viñedos pertenecientes a parcelas colindantes. En algunas imágenes, dichas hileras aparecían tan próximas a la viña principal que podían interferir en el análisis, afectando tanto a la evaluación del modelo como a la extracción precisa de parámetros agronómicos de la parcela de interés. Esta tarea de preprocesamiento fue realizada en colaboración con un compañero que comparte la misma tutoría de TFG.

Para ilustrar este proceso y su justificación, se presenta a continuación un ejemplo gráfico:

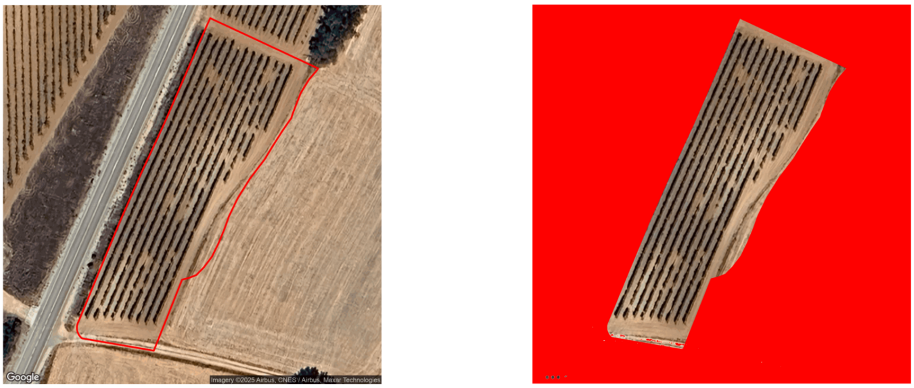


Figura 4.3: Ejemplo imagen original vs preprocesada para test

Para garantizar una evaluación fiable del modelo y prevenir el sobreajuste, el conjunto de datos se dividió de la siguiente forma:

- 80 % de los datos para entrenamiento.
- 20 % para validación.

La partición se ha realizado utilizando la función *train_test_split* de la librería *scikit-learn*, con semilla fija (*random_state*=42) para asegurar la reproducibilidad de los experimentos. Esta división se aplica sobre los índices de los archivos, asegurando que no haya solapamiento entre imágenes de entrenamiento y prueba.

Antes de ser procesadas por el modelo, todas las imágenes y máscaras han sido sometidas a las siguientes transformaciones:

- **Redimensionado** a 1024 × 1024 píxeles.
- **Conversión a tensores** mediante *torch.tensor*, normalizando los valores de píxel en el rango [0, 1].
- **Transposición del orden de canales** (de H×W×C a C×H×W), para compatibilidad con *PyTorch*.
- **Binarización de máscaras**, convirtiéndolas a *float32* con una sola dimensión (1 canal).

Este *dataset* tiene algunas limitaciones:

- El tamaño del *dataset* es limitado, lo que puede afectar la generalización del modelo.
- La anotación manual, aunque precisa, puede introducir sesgo humano o pequeñas inconsistencias.
- La diversidad geográfica y temporal está presente, pero no es exhaustiva; por tanto, el modelo se ajusta mejor a regiones con características similares a las del conjunto original.
- No se han utilizado imágenes multiespectrales (como las de *Sentinel-2*), por lo que el sistema se limita a análisis en el espectro visible (RGB).

A pesar de las limitaciones en tamaño y variedad, su diseño cuidado y su enfoque en el problema concreto de la segmentación de hileras de viñedo lo convierten en una base sólida y representativa para validar el funcionamiento del modelo implementado.

4.1.2. Hiperparámetros

Los hiperparámetros son parámetros externos al modelo que no se aprenden durante el entrenamiento, pero que influyen directamente en el comportamiento del proceso de aprendizaje. A diferencia de los parámetros internos (como los pesos de las neuronas), los hiperparámetros deben establecerse antes de iniciar el entrenamiento y determinan cómo se entrena el modelo, con qué velocidad, durante cuánto tiempo y bajo qué condiciones.

La correcta elección y ajuste de estos hiperparámetros es fundamental para alcanzar un buen rendimiento del modelo, evitar problemas como el sobreajuste o el estancamiento, y garantizar una buena generalización del sistema ante datos nuevos.

A continuación se describen los hiperparámetros más comunes utilizados en el entrenamiento de modelos de aprendizaje profundo, los cuales ha sido usados en este caso.

- **Tasa de aprendizaje (*learning rate*)**
  La tasa de aprendizaje es uno de los hiperparámetros más importantes. Controla el tamaño del paso que da el modelo al actualizar sus pesos después de cada iteración. Un valor demasiado alto puede hacer que el modelo no aprenda o diverja, mientras que uno demasiado bajo puede hacer que el entrenamiento sea extremadamente lento o que se quede atrapado en mínimos locales.

- **Función de pérdida (*loss function*)**
  La función de pérdida mide la diferencia entre la salida del modelo y el valor esperado. Es el valor que el modelo trata de minimizar durante el entrenamiento. La elección de la función de pérdida depende del tipo de problema (clasificación, regresión, segmentación, etc.). En problemas de segmentación binaria, suele usarse la entropía cruzada o combinaciones con métricas específicas como el coeficiente Dice o el *IoU*.

- **Optimizador**
  El optimizador es el algoritmo encargado de actualizar los pesos del modelo en función del valor de la pérdida y de su gradiente. Existen distintos tipos de optimizadores, como *SGD, Adam, RMSProp*, entre otros. Cada uno tiene sus propias características y parámetros internos, como *momentum* o tasas de aprendizaje adaptativas.

- **Tamaño de lote (*batch size*)**
  El tamaño de *batch* o lote indica cuántas muestras se utilizan en cada iteración para calcular el gradiente y actualizar los pesos. *Batches* grandes tienden a proporcionar estimaciones más estables del gradiente, mientras que *batches* pequeños pueden introducir más ruido pero también ayudan a escapar de mínimos locales. Este parámetro también está condicionado por la memoria disponible, especialmente cuando se entrena en GPU.

- **Número de épocas (*epochs*)**
  Una época representa una pasada completa por el conjunto de entrenamiento. El

número de épocas define cuántas veces el modelo ve cada muestra. Un número muy bajo puede conducir a un modelo que no ha aprendido lo suficiente (subentrenamiento), mientras que un número excesivo puede llevar al sobreajuste si no se aplica ningún mecanismo de control.

- ***Early stopping* (parada temprana)**
  El *early stopping* es una técnica que detiene automáticamente el entrenamiento cuando el modelo deja de mejorar en el conjunto de validación tras un número determinado de épocas. Permite evitar el sobreajuste y reducir el tiempo de entrenamiento innecesario. Suele configurarse con parámetros como "paciencia" (número de épocas sin mejora) y "delta" (mejora mínima considerada relevante).

- **Tamaño de entrada (*input size*)**
  Aunque no suele considerarse un hiperparámetro clásico, el tamaño de las imágenes de entrada afecta directamente al rendimiento del modelo y a los requisitos computacionales. Imágenes más grandes conservan más detalle, pero también incrementan el tiempo de entrenamiento y la memoria requerida.

- **Funciones de activación, regularización y arquitectura**
  Otros aspectos configurables que, aunque a veces se consideran parte del diseño del modelo, también pueden tratarse como hiperparámetros:

  - **Funciones de activación** (*ReLU, Sigmoid*, etc.): afectan la capacidad de representar relaciones no lineales.
  - **Regularización** (*Dropout, L2/L1*): ayudan a evitar el sobreajuste.
  - **Número de capas, filtros y neuronas**: definen la capacidad del modelo y también influyen en el tiempo de entrenamiento y riesgo de sobreajuste.

4.2. Solución

Antes de llegar a la solución definitiva, se evaluaron distintas arquitecturas que, en principio, podrían haber sido adecuadas para el problema planteado. En una primera fase, se probaron variantes del modelo *YOLO*, tanto en su versión orientada a la detección mediante cajas delimitadoras rotadas (*OBB*) como en su versión para segmentación semántica. Este enfoque se consideró inicialmente viable, ya que una detección precisa de las hileras podría bastar para estimar su longitud. Sin embargo, los resultados obtenidos con *YOLO* fueron notablemente inferiores a los logrados posteriormente con *U-Net*.

Este rendimiento limitado puede deberse a varias causas. En primer lugar, *YOLO* está optimizado para tareas de detección de objetos en imágenes naturales, con clases bien definidas y entrenado sobre grandes cantidades de datos anotados. En el contexto de nuestro proyecto, las hileras de viñedos no tienen una forma fija, su apariencia varía según la iluminación, el fondo y la resolución, y no se comportan como .ºbjetosçlásicos. Además, aunque se entrenaron diferentes versiones del modelo (como *yolo11m-seg, yolo11l-seg,*

yolo11m-obb, yolo11l-obb, entre otros), ninguno logró capturar con suficiente precisión la estructura fina y alargada de las filas.

También se exploró el uso de *DeepLabv3*, una arquitectura avanzada para segmentación semántica que incorpora módulos de dilatación espacial y conexiones multiescala. A pesar de su potencial, los resultados obtenidos seguían siendo inferiores a los proporcionados por *U-Net*. *DeepLabv3* fue desarrollado y optimizado principalmente sobre *datasets* como *PASCAL VOC* o *Cityscapes*, que contienen objetos bien definidos (coches, personas, edificios...). Estos entornos presentan clases fácilmente distinguibles y con estructuras espaciales más cerradas. Las hileras de viñedos, en cambio, son estructuras alargadas, finas y con textura repetitiva, lo que puede no encajar bien con los supuestos de *DeepLabv3*.

Aunque *DeepLabv3* incluye módulos de dilatación (*atrous convolutions*) para captar contexto a múltiples escalas, tiende a perder detalle en estructuras delgadas o lineales. Las filas de viñedos tienen precisamente ese patrón: estructuras finas que se diluyen si la red no tiene suficiente capacidad para conservar bordes estrechos. Además, *DeepLabv3* carece de mecanismos explícitos para mantener la coherencia a lo largo de una estructura lineal, algo que sí ofrece *U-Net* gracias a sus *skip connections*, las cuales preservan información de bajo nivel desde las primeras capas del *encoder*, mejorando así la precisión en los bordes.

En una fase posterior del análisis, se consideraron arquitecturas aún más modernas como *SegFormer* y *TransUNet*, ambas basadas en mecanismos de atención (*Transformers*) que, en teoría, permiten modelar relaciones espaciales de largo alcance y han demostrado un rendimiento destacado en *benchmarks* genéricos de segmentación. No obstante, varios factores desaconsejaron su adopción en este caso:

- **Requisitos computacionales elevados**: tanto *SegFormer* como *TransUNet* requieren mayor capacidad de memoria y potencia de cálculo que *U-Net*, lo que dificulta su entrenamiento en entornos limitados, como los disponibles para este proyecto.
- **Necesidad de grandes volúmenes de datos**: los *Transformers* suelen requerir grandes cantidades de datos para alcanzar un rendimiento óptimo. Dado que el conjunto de datos empleado es reducido y está etiquetado manualmente, estas arquitecturas no pudieron entrenarse eficazmente sin incurrir en sobreajuste.
- **Complejidad de implementación y ajuste**: frente a la simplicidad modular de *U-Net*, los modelos basados en *Transformers* implican una mayor complejidad arquitectónica y dependencia de preentrenamiento especializado, lo que dificulta su personalización en tareas específicas como la detección de hileras.

A pesar de estas limitaciones, se realizaron pruebas preliminares con versiones reducidas de *SegFormer-B0* y *TransUNet* preentrenadas, con los siguientes resultados aproximados sobre un subconjunto validado manualmente (resolución 512×512):

Modelo	*IoU* validación	Pérdida (*Loss*)	Observaciones
U-Net (ResNet34)	**0,8237**	**0,1152**	Mejor segmentación general, bordes definidos
SegFormer-B0	0,7812	0,1487	Mayor fragmentación en hileras finas
TransUNet (ViT-S)	0,7929	0,1396	Buen contexto global, peor en detalles locales
DeepLabv3+ (ResNet34)	0,8034	0,1278	Mejor que *TransUNet*, pero peor en continuidad

Tabla 4.2: Comparación preliminar entre modelos sobre un subconjunto de validación

Los resultados muestran que, si bien las arquitecturas basadas en *Transformers* ofrecen ventajas teóricas en el modelado del contexto global, no superan en la práctica, en este escenario concreto, el rendimiento de una *U-Net* bien ajustada. La arquitectura utilizada presenta además una excelente relación entre precisión, capacidad de generalización y eficiencia computacional.

Por todas estas razones, se optó finalmente por desarrollar una arquitectura basada en *U-Net*, adaptada específicamente a nuestro problema. Esta arquitectura, originalmente diseñada para segmentación biomédica, ha demostrado ser especialmente eficaz en tareas donde los objetos de interés tienen bordes difusos o estructuras irregulares. En nuestro caso, se utilizó una variante de *U-Net* con *encoder* preentrenado (como se describirá más adelante), lo cual permitió mejorar notablemente los resultados a pesar de trabajar con un conjunto de datos limitado y con anotaciones manuales imperfectas.

La solución propuesta en este Trabajo de Fin de Grado se basa en un modelo de segmentación semántica profunda, diseñado específicamente para identificar y extraer automáticamente las filas de viñedos presentes en imágenes aéreas o satelitales. Dado el carácter estructurado y repetitivo de estos cultivos en hileras, se optó por una arquitectura capaz de capturar detalles espaciales finos y mantener la contextualización global de la escena: una red *U-Net* con un *encoder ResNet34* preentrenado.

Esta combinación permite aprovechar la potencia del *transfer learning* en el *encoder* para extraer representaciones ricas y robustas, y al mismo tiempo reconstruir con precisión mapas de segmentación detallados a través de la estructura tipo U de la red. Pero antes de explicar esto más en detalle, se introducirá el problema desde el principio.

A diferencia de las tareas de clasificación, en las que se asigna una única etiqueta a una imagen completa, la segmentación semántica requiere etiquetar individualmente cada

píxel de la imagen. El resultado es un mapa de segmentación, es decir, una imagen de salida con la misma resolución que la de entrada, en la que cada píxel ha sido sustituido por la clase a la que pertenece.

En las arquitecturas de redes neuronales convolucionales tradicionales, las sucesivas capas convolucionales y operaciones de *pooling* tienden a reducir progresivamente la resolución espacial de los mapas de características, con el objetivo de capturar representaciones más abstractas y profundas. Este enfoque es adecuado para tareas como la clasificación de imágenes, en las que la salida se obtiene mediante la conexión de las últimas capas convolucionales a un perceptrón multicapa (*MLP*). En el caso de la detección de objetos, se emplean salidas intermedias a distintas escalas para generar predicciones de cajas delimitadoras y etiquetas.

Sin embargo, en la segmentación semántica, es necesario recuperar la resolución original de la imagen para que el modelo pueda generar una predicción a nivel de píxel. Esto requiere diseñar arquitecturas específicas que integren mecanismos de *upsampling*, como capas deconvolucionales o interpolaciones, junto con estrategias de fusión de información de diferentes niveles de profundidad.

Una aproximación inicial al problema de la segmentación podría consistir en utilizar una red convolucional (CNN) diseñada de forma que mantenga constante la resolución espacial de los mapas de características a lo largo de todas sus capas. Esto puede lograrse mediante una configuración cuidadosa de los filtros y evitando el uso de operaciones de *pooling*.

Sin embargo, este tipo de arquitectura presenta limitaciones importantes. Por un lado, al no reducir la resolución, la red pierde la capacidad de capturar información a diferentes escalas, lo que resulta esencial para comprender contextos complejos dentro de la imagen. Por otro lado, mantener la resolución constante durante todo el procesamiento implica un aumento significativo del coste computacional, tanto en términos de memoria como de tiempo de entrenamiento.

Para abordar estos problemas, se recurre con frecuencia a arquitecturas de tipo *encoder-decoder*. En este enfoque, una primera etapa (*encoder*) se encarga de extraer características progresivamente más abstractas y profundas, mediante sucesivas reducciones de resolución. Posteriormente, una segunda etapa (*decoder*) se encarga de reconstruir las dimensiones espaciales originales de la imagen, generando una predicción detallada a nivel de píxel.

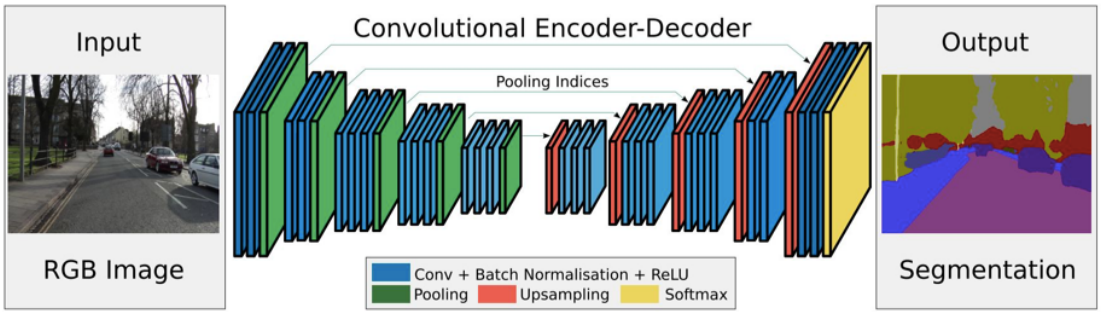


Figura 4.4: Arquitectura *encoder-decoder*

Uno de los mecanismos más utilizados para recuperar la resolución espacial en la fase de decodificación son las convoluciones transpuestas (*transposed convolutions*). Estas capas funcionan de forma análoga a las convoluciones tradicionales, pero en sentido inverso: en lugar de reducir el tamaño de los mapas de características, aprenden a expandirlos, aplicando filtros entrenables que permiten realizar un *upsampling* estructurado. Este tipo de capas permite que el modelo no solo recupere tamaño, sino que también aprenda cómo hacerlo de la manera más eficaz para mejorar la precisión de la segmentación.

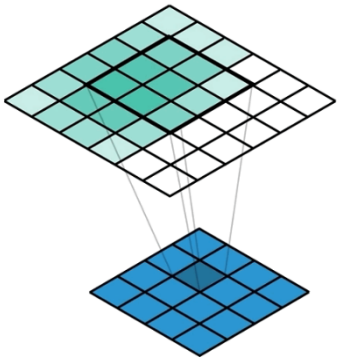


Figura 4.5: Ejemplo de convolución transpuesta

Este ejemplo muestra cómo utilizar una convolución transpuesta (`ConvTranspose2d`) en *PyTorch* para aumentar la resolución espacial de un mapa de características:

```
import torch

input = torch.randn(64, 10, 20, 20)
# aumentamos la dimension x2
conv_trans = torch.nn.ConvTranspose2d(
    in_channels=10,
    out_channels=20,
```

```
kernel_size=2,
stride=2)
```

Código 4.1: Código de ejemplo convolución transpuesta

Se crea un tensor *input* de ejemplo de tamaño (64, 10, 20, 20), que podríamos encontrar dentro de una red convolucional:

- **64**: tamaño del *batch* (número de imágenes).
- **10**: número de canales de entrada.
- **20 x 20**: dimensiones espaciales (alto y ancho).

Después se define una capa de convolución transpuesta (`ConvTranspose2d`) que:

- Toma **10** canales de entrada (este valor tiene que ser igual al número de canales de entrada del tensor *input*).
- Produce **20** canales de salida.
- Utiliza un filtro de tamaño **2×2**.
- Aplica una *stride* de **2**, lo cual duplica las dimensiones espaciales de la entrada.

El *stride*=2 y el *kernel* de 2×2 permiten que cada píxel de entrada "expanda" su información en una vecindad de 2×2 en la salida, aumentando así la resolución x2 en cada dimensión, por lo que el tensor resultante tendrá un tamaño de (64, 20, 40, 40).

Este enfoque permite diseñar arquitecturas más eficientes, capaces de capturar información relevante a múltiples escalas mediante la compresión progresiva de los mapas de características. No obstante, reconstruir con precisión los detalles espaciales a partir únicamente de la salida final del *encoder* puede resultar insuficiente o ineficaz, especialmente en tareas donde la localización precisa de los objetos es fundamental.

Para abordar este desafío, se desarrolló una de las arquitecturas más reconocidas y ampliamente utilizadas en segmentación de imágenes: la red *U-Net*. Esta arquitectura introduce un mecanismo de conexión entre etapas simétricas del *encoder* y del *decoder*, permitiendo recuperar información de bajo nivel con gran precisión espacial.

Una de las características distintivas de esta arquitectura es que, en cada etapa del *decoder*, no solo se utiliza la salida de la capa anterior, sino que también se incorpora la salida de la capa correspondiente del *encoder* mediante una conexión directa (*skip connection*). Esta estrategia permite a la red combinar información de alto nivel semántico con detalles espaciales de baja escala, lo que mejora significativamente la precisión de la segmentación y la capacidad de reconstrucción de bordes y estructuras finas (como es nuestro caso).

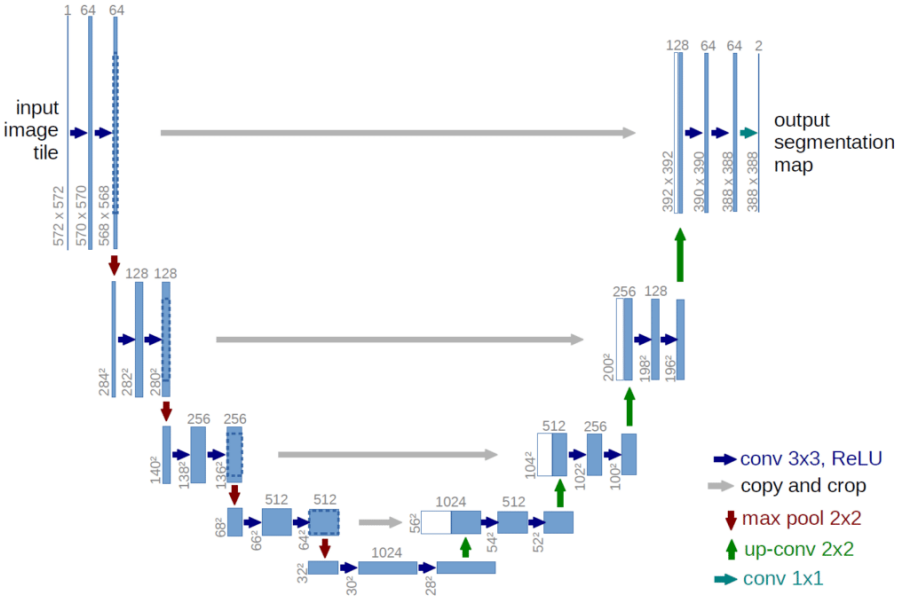


Figura 4.6: Arquitectura de la red *U-Net*

Antes de profundizar en la arquitectura de la red neuronal utilizada, es fundamental aclarar el formato que deben adoptar las imágenes y las máscaras que alimentarán el modelo durante el entrenamiento. Asegurar la correcta preparación de estos datos es crucial para el buen funcionamiento de la red y para evitar errores durante la fase de entrenamiento o inferencia.

Las imágenes, originalmente en formato RGB y con valores de píxel en el rango [0, 255], deben ser transformadas en tensores flotantes normalizados con valores en el rango [0.0, 1.0]. Para ello, se realiza una conversión a tipo *float32* y una normalización dividiendo por 255, lo que garantiza una escala de entrada coherente con los requisitos de las redes neuronales profundas. Además, dado que *PyTorch* espera el orden de dimensiones como (canales, alto, ancho), es necesario transponer la imagen desde su formato original (alto, ancho, canales) a (canales, alto, ancho).

En cuanto a las máscaras, que representan la clase asignada a cada píxel (en este caso, viñedo o no viñedo), se parte de imágenes en escala de grises. Estas se binarizan mediante un umbral (por ejemplo, valores superiores a 127 se consideran clase positiva), generando una matriz de valores 0.0 y 1.0. Al igual que en el caso de las imágenes, también se les añade un canal adicional para obtener el formato esperado (1, alto, ancho), y se convierten a tensores de tipo *float32*.

Este proceso de preprocesamiento garantiza que tanto las imágenes como las máscaras tengan el formato, tipo de dato y estructura dimensional adecuada para ser procesadas por el modelo de segmentación implementado, evitando inconsistencias y favoreciendo la correcta propagación de la información a lo largo de la red.

El siguiente código ha sido el utilizado en en este caso para ello:

```
binary_mask = ((mask > 127).astype(np.float32))[..., np.newaxis]

image = torch.tensor(image.transpose(2, 0, 1), dtype=torch.float32)
        / 255.0

binary_mask = torch.tensor(binary_mask.transpose(2, 0, 1),
              dtype=torch.float32)
```

Código 4.2: Código preprocesameinto imágenes y máscaras

A continuación, se describe en detalle la arquitectura de la red *U-Net*, apoyándose en su diagrama esquemático de la figura 4.6. Esta arquitectura se compone de dos bloques principales: un *encoder* (contracción) y un *decoder* (expansión), conectados entre sí mediante conexiones de salto (*skip connections*).

En el *encoder*, las flechas rojas del diagrama representan operaciones de *MaxPooling* 2×2, que reducen las dimensiones espaciales a la mitad en cada etapa. A su vez, las flechas azules corresponden a bloques de convoluciones 3×3 seguidas de activaciones *ReLU*, encargadas de extraer y refinar las características visuales. Con cada paso hacia abajo en el *encoder*, se duplica el número de filtros para capturar información de mayor complejidad. Este proceso se repite un número determinado de veces (en este caso, cinco etapas), comprimiendo progresivamente la información de la imagen.

En el *decoder*, se revierte este proceso. Las flechas verdes indican el uso de convoluciones transpuestas, que permiten realizar *upsampling* aprendiendo a recuperar la resolución espacial original. Además, en cada etapa del *decoder*, se concatenan las activaciones correspondientes del *encoder* a través de las conexiones de salto. Esto permite preservar información espacial fina y detallada que podría haberse perdido durante la fase de contracción.

Durante el proceso de expansión, el número de filtros se reduce progresivamente a la mitad en cada etapa, equilibrando la arquitectura de forma simétrica respecto al *encoder*.

Finalmente, la salida del modelo pasa por una convolución 1×1, cuya función es reducir el número de canales de salida al número de clases deseado (en este caso 2). Esta capa actúa de forma pixel a pixel, combinando la información presente en todos los canales para generar una máscara predicha del mismo tamaño que la imagen original. A esta capa se le aplica una función de activación *sigmoid*, que transforma la salida en valores entre 0 y 1, interpretables como probabilidades de pertenencia a las clases correspondientes.

Una primera implementación funcional de esta arquitectura puede estructurarse mediante una clase que reciba como parámetros el número de clases de salida (en este caso, 1 para segmentación binaria) y el número de canales de entrada (3, correspondientes a imágenes RGB). Esta clase puede estar acompañada por una lista que defina el número de filtros en cada etapa del *encoder* y del *decoder*, reflejando la simetría de la arquitectura *U-Net*. Por ejemplo, una configuración típica podría ser [16, 32, 64, 128] donde los valores se aplican en orden ascendente en el *encoder* y en orden descendente en el *decoder*.

Encoder

En la primera etapa, se define una capa inicial que recibe la imagen de entrada. Esta capa consiste en dos aplicaciones consecutivas de la función `conv3x3_bn`, que encapsula los siguientes elementos:

- Una convolución 2D con filtros de tamaño 3×3 y *padding*=1 (manteniendo las dimensiones espaciales).
- Una capa de *Batch Normalization*.
- Una función de activación *ReLU*.

La decisión de no alterar las dimensiones espaciales en esta etapa se debe a que la reducción de resolución se delega al *MaxPooling*, que se aplicará en las siguientes capas del *encoder*.

Las siguientes capas del *encoder* se implementan mediante una función auxiliar, `encoder_conv`, que se aplica de forma secuencial tres veces. Esta función realiza:

- Una operación de *MaxPool* 2×2, que reduce las dimensiones espaciales a la mitad.
- Dos bloques `conv3x3_bn`, que permiten refinar las características tras la compresión.

De este modo, el *encoder* queda definido como una secuencia de etapas que reducen progresivamente la resolución de la imagen y aumentan la profundidad de los mapas de características.

Decoder

El *decoder* se implementa mediante una función llamada `deconv`, que se aplica también tres veces de forma secuencial, utilizando los filtros definidos en la lista original pero en orden inverso.

Cada bloque del *decoder* realiza:

- Una convolución transpuesta (`ConvTranspose2d`) con *kernel* 2×2 y *stride* 2, que duplica las dimensiones espaciales del mapa de características, realizando así un *upsampling* aprendido.
- Una concatenación (*skip connection*) con la salida correspondiente del *encoder* en la misma etapa, lo que permite recuperar información espacial de bajo nivel.
- Dos nuevas aplicaciones de la función *conv3x3_ bn*, que refinan la información fusionada.

La lógica de las *skip connections* se implementa en la función *forward*, que en cada etapa del *decoder*:

- Realiza el *upsampling* de la salida de la etapa anterior.
- Ajusta el tamaño si es necesario.
- Concatena esta salida con la activación correspondiente del *encoder*.
- Aplica dos convoluciones 3×3 con *BatchNorm* y *ReLU* para procesar la información conjunta.

Finalmente, la arquitectura se completa con una convolución 1×1 con tantos filtros como clases de salida (en este caso, 1), encargada de producir la máscara final de segmentación. Esta capa transforma los mapas de características en una única imagen con un canal, donde cada píxel representa la probabilidad de pertenecer a la clase positiva (en nuestro caso sería fila de viñedo).

Una vez definidas todas las capas, el flujo de datos a través de la red se compone de:

1. Aplicación secuencial de todas las capas del *encoder*, almacenando las salidas intermedias necesarias para las *skip connections*.
2. Aplicación secuencial de las capas del *decoder*, que reciben como entrada tanto la salida anterior como la correspondiente salida del *encoder*.
3. Aplicación de la capa final de salida, que devuelve la máscara predicha por el modelo.

Una vez definida la arquitectura del modelo, el siguiente paso consiste en entrenarlo utilizando un conjunto de imágenes y sus respectivas máscaras como supervisión. Para ello, se emplea la función de pérdida *BCEWithLogitsLoss*, especialmente diseñada para problemas de clasificación binaria por píxel. Esta función combina internamente una activación *sigmoid* sobre la salida del modelo (transformando los valores en probabilidades entre 0 y 1) con el cálculo de la entropía cruzada binaria, lo que permite comparar de forma eficiente la predicción con la máscara real.

Como algoritmo de optimización se utiliza *Adam*, un optimizador adaptativo ampliamente utilizado en redes neuronales profundas. Este recibe los parámetros del modelo y el valor definido para la tasa de aprendizaje (*learning rate*), ajustando de forma dinámica los pesos de la red en función del gradiente estimado.

El proceso de entrenamiento se lleva a cabo durante un número determinado de épocas (*epochs*). En cada una de ellas, se realiza el siguiente ciclo:

1. Se pasa una imagen de entrada al modelo para obtener una predicción.
2. Se calcula la función de pérdida comparando la predicción con la máscara real.
3. Se realiza la propagación hacia atrás (*backpropagation*) para obtener los gradientes.
4. Se actualizan los pesos del modelo mediante el optimizador.

Este proceso se repetiría iterativamente para todos los lotes del conjunto de entrenamiento, con el objetivo de minimizar la pérdida y mejorar progresivamente la capacidad del modelo para segmentar correctamente las filas de viñedo.

Durante el entrenamiento del modelo, un descenso progresivo en el valor de la función de pérdida puede indicar que la red está aprendiendo adecuadamente. Sin embargo, la pérdida por sí sola no es suficiente para evaluar de forma cuantitativa y directa la calidad de las segmentaciones generadas. Para ello, es necesario recurrir a una métrica de evaluación específica que mida el grado de similitud entre las máscaras predichas por el modelo y las máscaras reales.

Una de las métricas más utilizadas en tareas de segmentación semántica es el Índice de Intersección sobre Unión (IoU), de la que ya hemos hablado. Esta métrica calcula la proporción entre el área de intersección y el área de unión entre la predicción y la verdad de terreno, proporcionando un valor entre 0 y 1, donde 1 indica una coincidencia perfecta.

Para implementar esta métrica, se define una función que recibe como entrada la salida de la red y las máscaras reales. La salida del modelo, al no estar normalizada, se procesa primero con una función *sigmoid* para convertirla en valores de probabilidad entre 0 y 1. Posteriormente, se aplica un umbral (por ejemplo, 0.5) para binarizar la predicción y transformarla en una máscara segmentada.

Una vez binarizadas ambas máscaras, se calcula la intersección como el número de píxeles en los que ambas coinciden (predicción y etiqueta con valor 1), y la unión como el número de píxeles que están etiquetados como positivos en al menos una de las dos máscaras. Para evitar divisiones por cero, se añade una pequeña constante (epsilon) al numerador y denominador. El cociente resultante representa el valor de IoU.

Durante el entrenamiento del modelo, a medida que la función de pérdida disminuye, se espera que el valor de la métrica IoU aumente progresivamente. Esto indicaría que las predicciones del modelo se ajustan cada vez mejor a las máscaras reales, lo que refleja una mejora en la calidad de la segmentación.

Aunque la arquitectura descrita permite obtener resultados satisfactorios en la tarea de segmentación, es posible mejorar su rendimiento aplicando técnicas de aprendizaje por transferencia (*transfer learning*). Esta estrategia consiste en reutilizar los conocimientos adquiridos por un modelo previamente entrenado sobre un gran conjunto de datos (por ejemplo, *ImageNet*), en lugar de entrenar la red desde cero.

En este caso, se sustituye el *encoder* original de la red por las capas convolucionales de una red *ResNet*, una arquitectura más profunda y robusta, ampliamente validada en tareas de visión por computador. Al incorporar una *ResNet* preentrenada, el modelo puede aprovechar representaciones de alto nivel ya aprendidas, lo que permite obtener mejores resultados con menos datos y en menos épocas de entrenamiento.

Este enfoque permite combinar la capacidad de representación de una red profunda como *ResNet* en la fase de extracción de características, con la estructura del *decoder* de *U-Net* para recuperar la resolución y generar las máscaras de segmentación. El resultado es una arquitectura más eficiente y precisa, especialmente útil en contextos con conjuntos de datos limitados.

A continuación se detalla cómo se implementa la arquitectura *U-Net* con un *encoder*

basado en *transfer learning* utilizando *ResNet34*. La implementación sigue una estructura muy similar a la arquitectura *U-Net* estándar descrita anteriormente, con la principal diferencia en la sustitución del *encoder* por una red *ResNet* preentrenada.

El primer paso consiste en reemplazar el *encoder* tradicional por las capas convolucionales de la red *ResNet34*, la cual ha sido previamente entrenada sobre un gran conjunto de datos (por ejemplo, *ImageNet*). Dado que *ResNet* está diseñada para trabajar con imágenes RGB de 3 canales, si se desea utilizar imágenes en escala de grises (1 canal), es necesario redefinir la primera capa convolucional. Esta capa debe mantenerse con las mismas características (*kernel, stride y padding*), pero modificando el número de canales de entrada para que sea compatible con el nuevo tipo de imágenes.

Con esta modificación, el *encoder* queda completamente definido utilizando los bloques residuales de *ResNet*. En cuanto al *decoder*, su implementación se mantiene idéntica a la de la arquitectura *U-Net* anterior: se utilizan bloques de *upsampling* mediante convoluciones transpuestas, combinados con *skip connections* que permiten recuperar información espacial desde las capas intermedias del *encoder*.

La única consideración importante en el *decoder* es que los números de filtros utilizados por *ResNet* deben conocerse, ya que estos definen el tamaño de entrada y salida de cada bloque de deconvolución. Estos filtros se deben aplicar en orden inverso durante la etapa de expansión.

En caso de querer emplear una variante distinta de *ResNet* (como *ResNet18* o *ResNet50*), es importante tener en cuenta que la estructura interna y el número de filtros cambian, lo que implicaría ajustes adicionales en el *decoder* para mantener la coherencia dimensional en la arquitectura.

4.3. Entrenamiento y generación de máscaras

El proceso de entrenamiento de un modelo de aprendizaje profundo no se limita únicamente a su arquitectura, sino que depende en gran medida de la correcta elección de los hiperparámetros, así como de la estrategia de entrenamiento seguida. Estos aspectos condicionan el comportamiento del modelo durante el aprendizaje y determinan en última instancia su capacidad para generalizar sobre nuevos datos.

En esta sección se describen las distintas fases llevadas a cabo para alcanzar un modelo óptimo. En primer lugar, se presenta el proceso de ajuste de hiperparámetros, en el cual se compararon diferentes configuraciones para identificar aquellas que ofrecían el mejor equilibrio entre precisión y eficiencia. Posteriormente, se detalla el entrenamiento definitivo del modelo seleccionado..

4.3.1. Ajuste de hiperparámetros

Antes de proceder al entrenamiento definitivo del modelo, fue necesario realizar un proceso de ajuste de hiperparámetros con el objetivo de maximizar el rendimiento de la arquitectura *U-Net* con *encoder ResNet34*. Este proceso consistió en la experimenta-

ción con distintas configuraciones de los parámetros que controlan el comportamiento del modelo durante el aprendizaje.

Los hiperparámetros evaluados incluyeron:

- **Tasa de aprendizaje (*learning rate*)**: se probaron distintos valores en el orden de magnitud entre 10^{-2} y 10^{-5} observando la estabilidad del entrenamiento y la convergencia de las métricas, concluyendo que una tasa de $0{,}0003(3e-4)$ ofrecía una convergencia progresiva y estable, sin oscilaciones excesivas en la función de pérdida.
- ***Batch size***: se probaron tamaños de 2, 4, 8 y 16, determinando que un tamaño de 4 ofrecía un equilibrio adecuado entre estabilidad y capacidad computacional, dadas las limitaciones del entorno de ejecución.
- **Número de épocas (*epochs*)**: se fijó un máximo de 100, con mecanismo de parada temprana (*early stopping*) para evitar el sobreentrenamiento.
- ***Patience* del *early stopping***: se probaron valores de paciencia entre 5 y 15, concluyendo que 10 permitía una buena tolerancia a fluctuaciones sin ralentizar el entrenamiento innecesariamente.
- **Resolución de entrada**: dado que el tamaño de las imágenes afecta tanto a la precisión de la segmentación como al coste computacional, se evaluaron tres resoluciones: 256×256, 512×512 y 1024×1024 píxeles. Las pruebas mostraron que a mayor resolución se obtenía mayor detalle en la segmentación. Sin embargo, el uso de imágenes de 1024×1024 superaba las capacidades computacionales disponibles, tanto en tiempo de entrenamiento como en consumo de memoria, lo que motivó finalmente la elección de 512×512 como tamaño de entrada para todo el proceso.

4.3.2. Entrenamiento y generación de máscaras

Una vez finalizado el proceso de ajuste de hiperparámetros, se procedió al entrenamiento del modelo definitivo con la configuración que demostró ofrecer el mejor rendimiento global, tanto en términos de precisión como de estabilidad y eficiencia computacional.

Cabe destacar que, como parte del proceso experimental, se realizaron múltiples entrenamientos con diferentes variantes del *dataset*, resoluciones, divisiones de entrenamiento/validación y configuraciones de red. Esta experimentación permitió comparar el comportamiento del modelo en distintos contextos, analizando cómo afectaban las variaciones en los datos o en la arquitectura a las métricas clave.

Tras esta fase comparativa, se seleccionó como entrenamiento principal aquel que ofreció los mejores resultados de segmentación, manteniendo una buena capacidad de generalización y un rendimiento computacional asumible. Esta configuración final utilizó los siguientes valores de hiperparámetros:

- **Tasa de aprendizaje**: 0.0003

- ***Batch size***: 4
- **Resolución de entrada**: 512×512
- **Épocas máximas**: 100
- ***Early stopping***: activado, con paciencia de 10 épocas
- **Función de pérdida**: *BCEWithLogitsLoss*
- **Optimizador**: *Adam*

El modelo fue entrenado utilizando un 80 % del conjunto de datos para entrenamiento y un 20 % para validación, seleccionado aleatoriamente con semilla fija para asegurar la reproducibilidad. La arquitectura empleada fue una *U-Net* con *encoder ResNet34* preentrenado, adaptada a la tarea de segmentación binaria de hileras de viñedo.

Durante el entrenamiento, se monitorizó de forma continua la función de pérdida y la métrica *IoU* sobre ambos conjuntos (entrenamiento y validación). El proceso se interrumpió automáticamente en la época 57 mediante el mecanismo de *early stopping*, al no observarse mejoras en la métrica de validación durante diez épocas consecutivas.

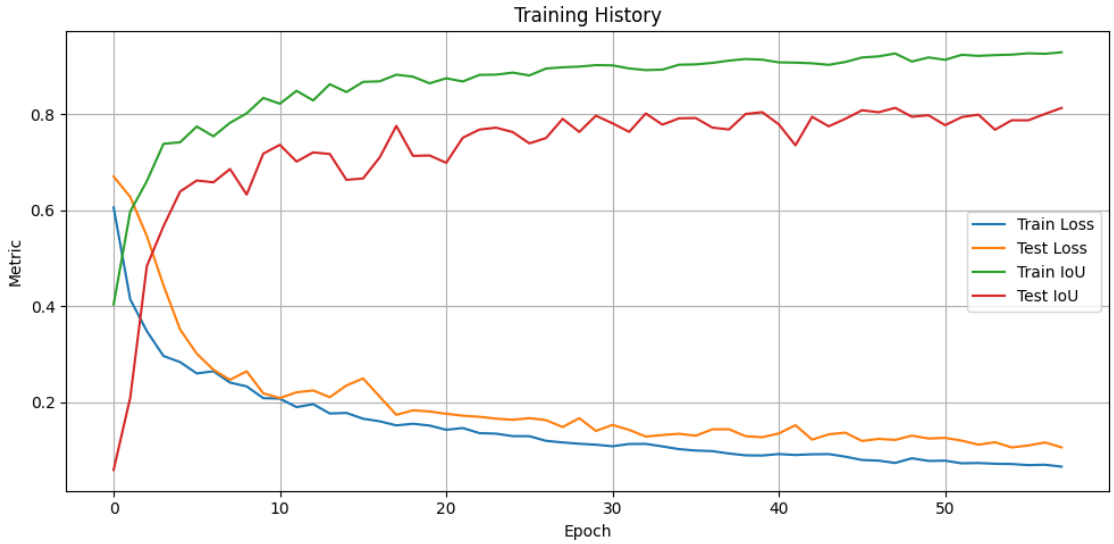


Figura 4.7: Evolución de las diferentes métricas en el entrenamiento

Los resultados obtenidos fueron los siguientes:

- ***Train IoU***: 0.9146
- ***Test IoU***: 0.8237
- ***Train Loss***: 0.0787

- ***Test Loss***: 0.1152

Estas métricas indican que el modelo fue capaz de aprender de forma efectiva las estructuras características de las hileras de viñedo, manteniendo un rendimiento elevado incluso sobre datos no vistos durante el entrenamiento. La diferencia moderada entre los resultados de entrenamiento y test sugiere una buena capacidad de generalización, sin evidencias de sobreajuste.

En la Figura 5.3 se muestra la evolución de las métricas a lo largo de las épocas. Las curvas reflejan una tendencia clara de descenso en la pérdida y de mejora en la IoU durante las primeras etapas del entrenamiento, estabilizándose progresivamente hasta la activación del *early stopping*.

Una vez entrenado el modelo, se procedió a evaluar su rendimiento cualitativo mediante la visualización de resultados sobre imágenes del conjunto de test. En esta etapa se comparan las máscaras predichas por el modelo con las máscaras de referencia, previamente generadas de forma manual.

Durante este análisis se observó que, en numerosos casos, las predicciones generadas automáticamente por el modelo presentaban un nivel de detalle y precisión superior al de las máscaras manuales, especialmente en los contornos y la continuidad de las hileras de viñedo. Este fenómeno se explica por la capacidad del modelo de generalizar patrones aprendidos durante el entrenamiento, aplicándolos de manera consistente incluso en regiones donde la anotación humana era ambigua o imprecisa.

Para ilustrar esta situación, se presentan a continuación varios ejemplos:

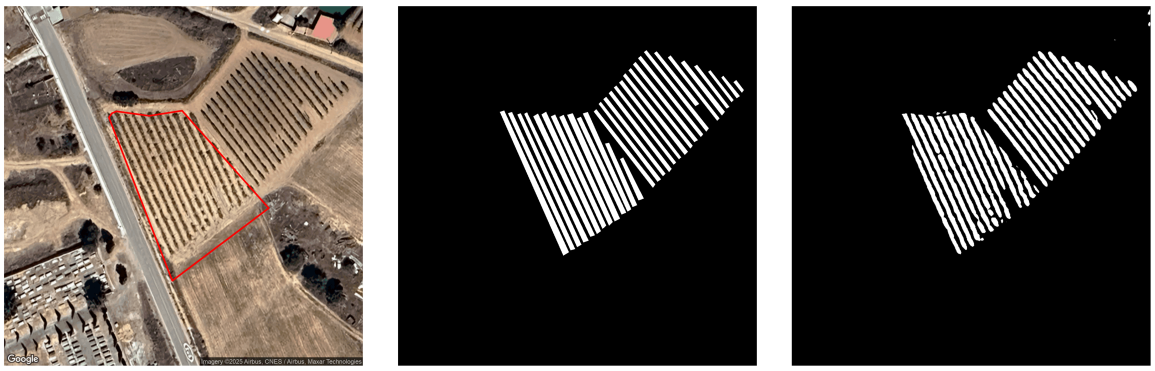


Figura 4.8: Comparación mascara anotada manualmente vs mascara predicha por el modelo

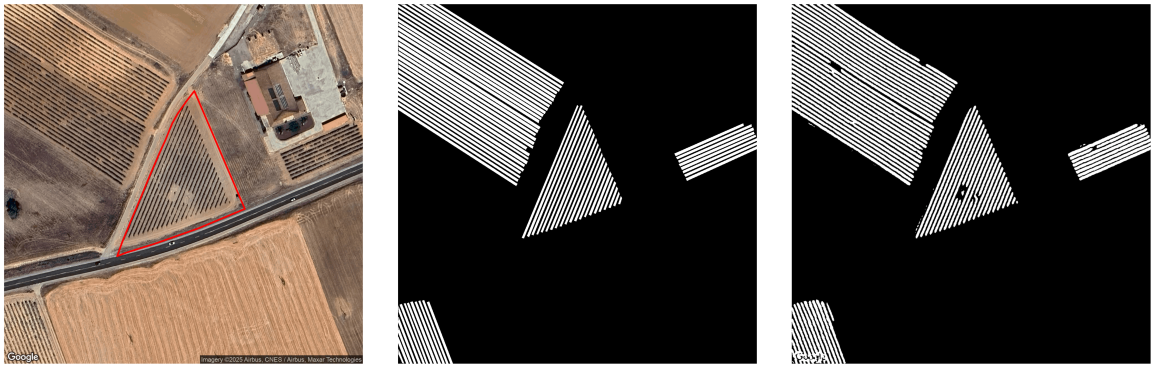


Figura 4.9: Comparación mascara anotada manualmente vs mascara predicha por el modelo

En el siguiente ejemplo se observa cómo el modelo es capaz de identificar casi por completo ciertas hileras de viñedo que no fueron incluidas en la anotación manual:

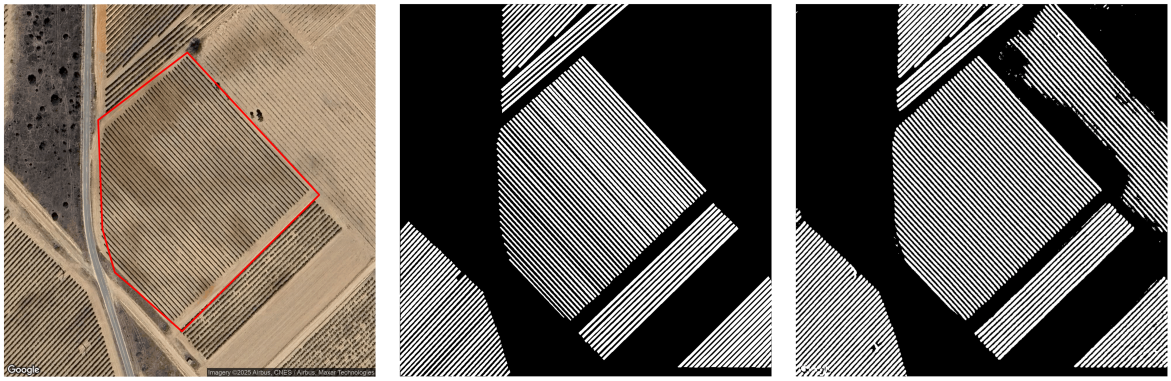


Figura 4.10: Comparación mascara anotada manualmente vs mascara predicha por el modelo

A continuación se presentan ejemplos de imágenes preprocesadas pertenecientes al conjunto de test, las cuales no fueron utilizadas durante el entrenamiento del modelo:

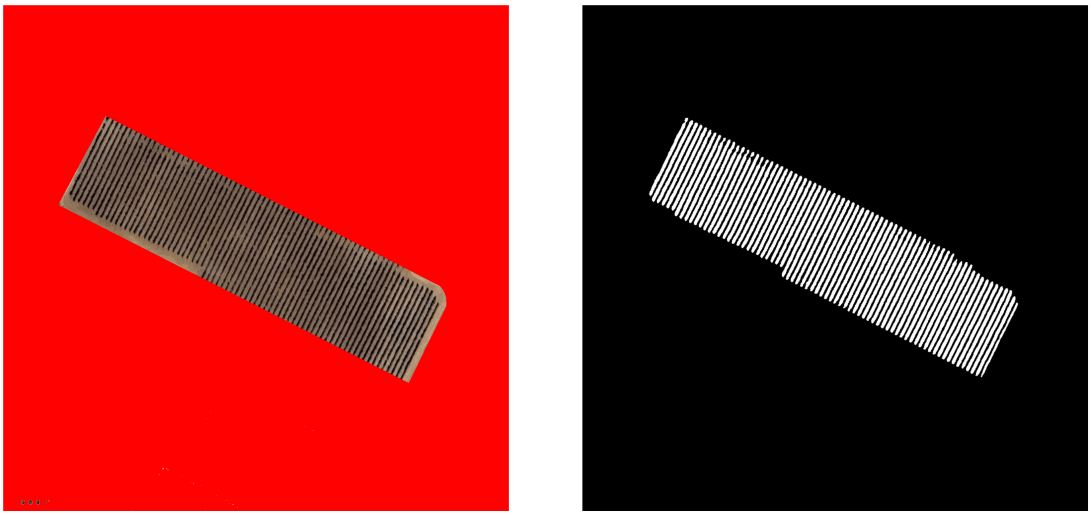


Figura 4.11: Máscara predicha para una imagen test preprocesada

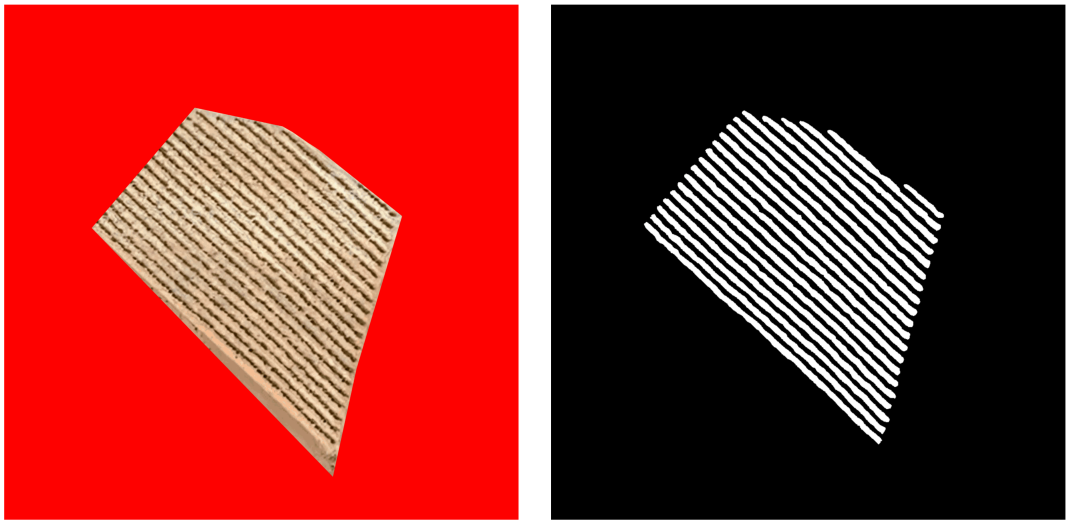


Figura 4.12: Máscara predicha para una imagen test preprocesada

Estas visualizaciones permiten constatar visualmente cómo el modelo no solo reproduce fielmente la geometría de las hileras, sino que en muchos casos corrige errores o lagunas presentes en las etiquetas de referencia, lo que sugiere que el sistema es capaz de ofrecer una segmentación más coherente y homogénea.

Aunque, en ciertas situaciones, la presencia de sombras o zonas oscuras en la imagen

dificulta que el modelo identifique correctamente las hileras, en contraste con los casos en los que la iluminación es favorable:

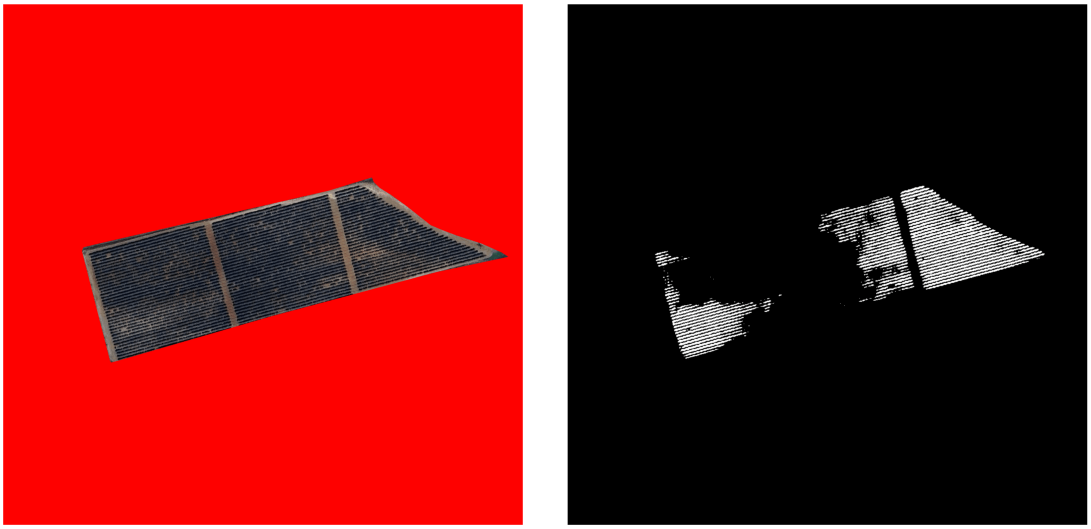


Figura 4.13: Máscara predicha para una imagen test preprocesada con baja luminosidad

En otros casos, la falta de resolución o un nivel de zoom insuficiente dificultan la segmentación, ya que impiden distinguir con claridad la separación entre las hileras:

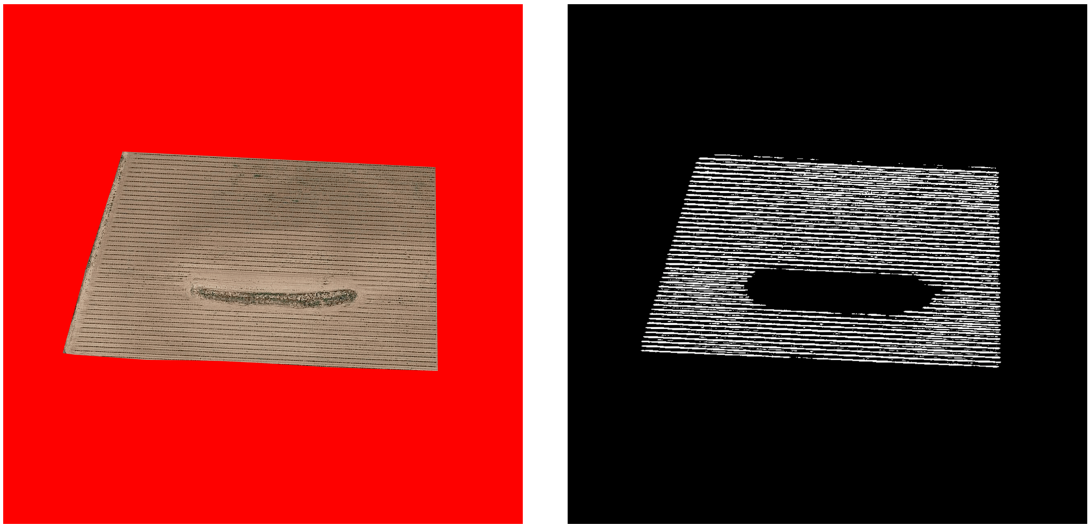


Figura 4.14: Máscara predicha para una imagen test preprocesada con un nivel de zoom insuficiente

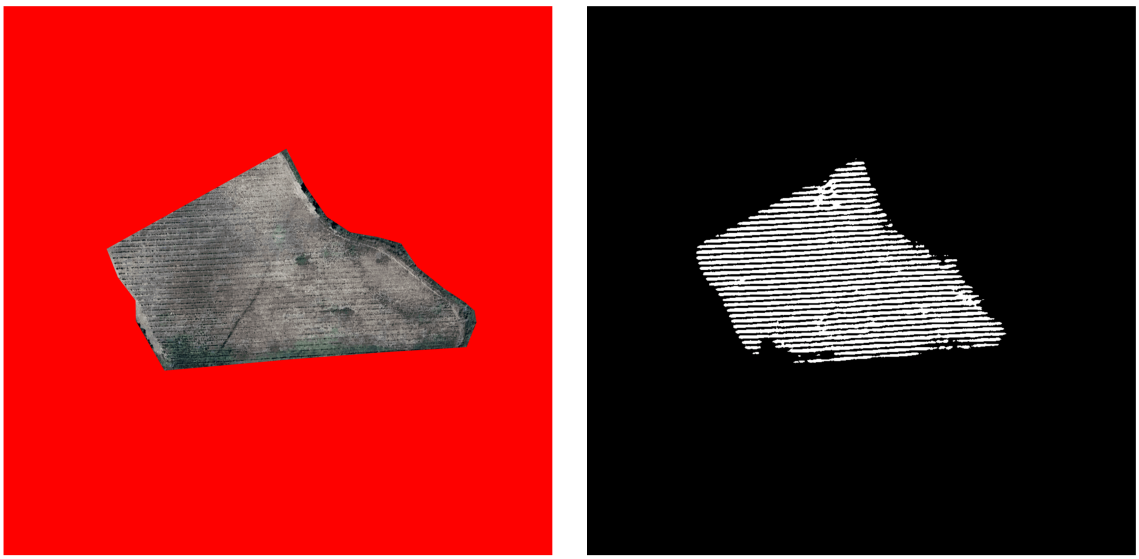


Figura 4.15: Máscara predicha para una imagen test preprocesada con un nivel de zoom insuficiente y baja luminosidad

Capítulo 5

Evaluación

La evaluación del modelo propuesto es una etapa fundamental para validar su efectividad y utilidad. En este capítulo se presentan los criterios utilizados para valorar el rendimiento del sistema, así como una reflexión crítica sobre los resultados obtenidos y propuestas prácticas para abordar sus limitaciones.

5.1. Métrica *IoU*

Aunque ya se ha mencionado anteriormente, se hace un repaso a esta métrica clave para contextualizar adecuadamente los resultados.

Para cuantificar la precisión de las predicciones del modelo, se ha utilizado como métrica principal el Índice de Intersección sobre Unión (IoU), una medida ampliamente aceptada en el ámbito de la segmentación semántica. Esta métrica se define como la relación entre el número de píxeles coincidentes (intersección) entre la predicción y la máscara de referencia, y el número total de píxeles etiquetados como positivos en al menos una de ellas (unión).

Si llamamos A la predicción y B a la anotación real:

$$IoU = \frac{A \cap B}{A \cup B}$$

De la definición podemos ver que $0 \leq IoU \leq 1$, por lo tanto, cuanto más se acerque a 1, la predicción es más certera. Sin embargo, en el contexto de este proyecto, el valor ideal de IoU no debe interpretarse necesariamente como 1, ya que las máscaras manuales utilizadas como referencia no siempre representan con precisión la realidad.

5.2. Validación cruzada geográfica

Con el objetivo de evaluar la capacidad general del modelo frente a distintas condiciones geográficas y visuales, se ha implementado una validación cruzada geográfica estratificada en tres zonas diferenciadas del *dataset*: A, B y C. La asignación de imágenes

a cada zona se ha realizado manualmente, en función de sus características visuales y del nivel de dificultad que presentan para la segmentación:

- **Zona A**: imágenes de parcelas con entorno limpio, sin elementos visuales interferentes. Se considera la zona más regular y sencilla.

- **Zona B**: incluye imágenes con un nivel medio de dificultad, debido a la presencia de caminos, edificios o cultivos adyacentes.

- **Zona C**: engloba imágenes especialmente difíciles, con sombras marcadas, exposición desigual o colores atípicos.

Para llevar a cabo esta validación cruzada, se han realizado tres entrenamientos independientes, en cada uno de los cuales se reserva una de las zonas como conjunto de test, utilizando las otras dos para el entrenamiento. De este modo, se evalúa la capacidad de generalización del modelo frente a distintos tipos de escenarios.

Los resultados obtenidos muestran una clara correlación entre la complejidad de la zona y el rendimiento del modelo. Concretamente:

- **Zona A como test**: el modelo alcanza su mejor rendimiento, con un IoU de 0,66342, lo que confirma que en escenarios regulares la segmentación es altamente precisa.

- **Zona B como test**: se obtiene un IoU de 0,59628, ligeramente inferior, reflejando la mayor complejidad visual de estas imágenes.

- **Zona C como test**: el rendimiento disminuye notablemente, alcanzando un IoU de 0,48622, lo cual era esperable dada la dificultad intrínseca de esta zona.

Estos resultados se representan en las siguientes gráficas de evolución durante el entrenamiento:

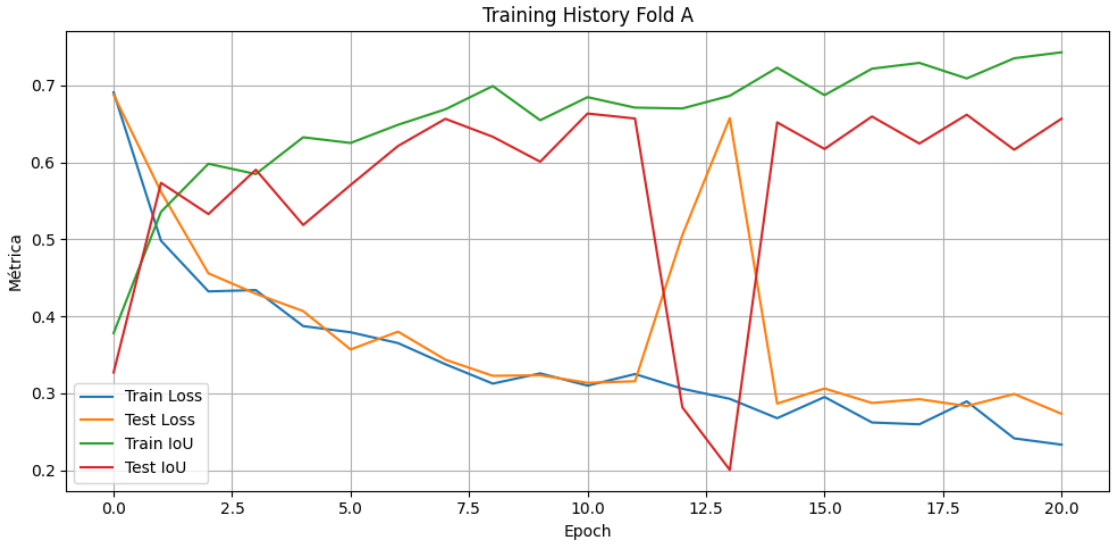


Figura 5.1: Evolución de las diferentes métricas en el entrenamiento utilizando el conjunto A como test

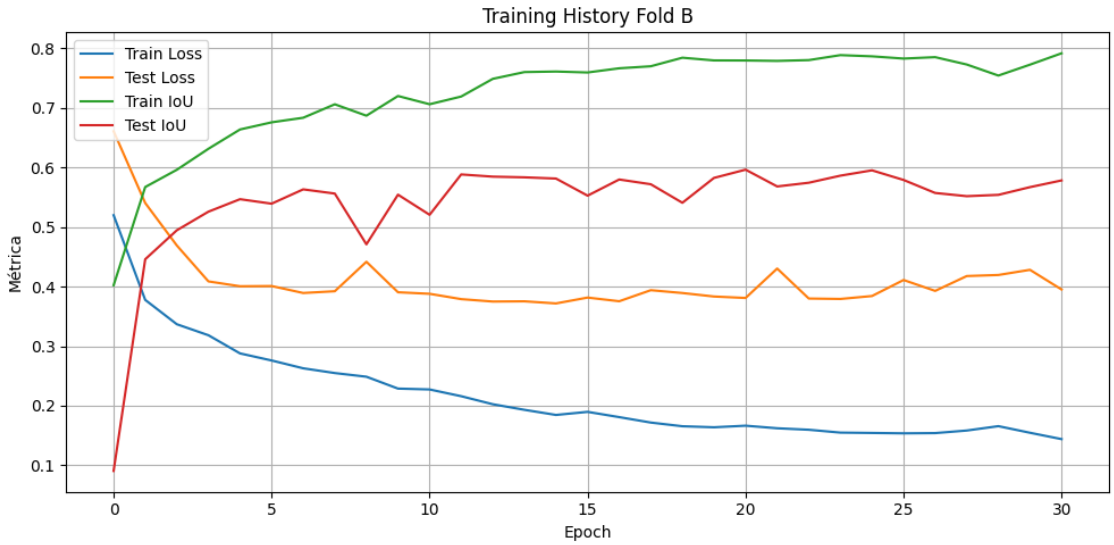


Figura 5.2: Evolución de las diferentes métricas en el entrenamiento utilizando el conjunto B como test

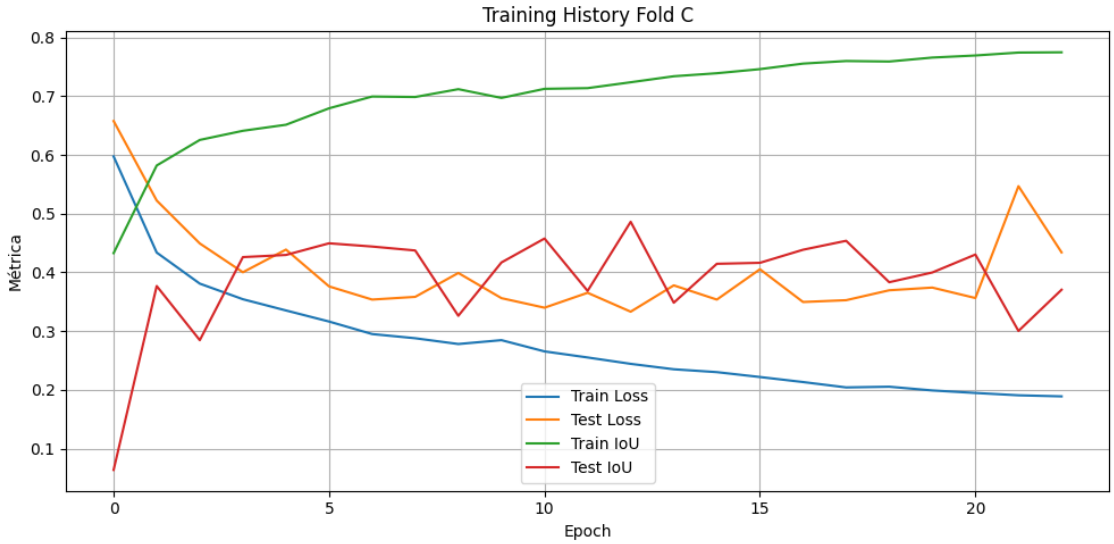


Figura 5.3: Evolución de las diferentes métricas en el entrenamiento utilizando el conjunto C como test

Este enfoque de validación cruzada permite no solo estimar el rendimiento medio del modelo, sino también analizar su comportamiento ante condiciones diversas, identificando con claridad los casos donde se obtiene una segmentación robusta y aquellos en los que se requiere mejora.

En los casos más complejos (particularmente en la zona C) ejemplificados en la Sección 4.3.2, se ha observado que el modelo falla principalmente por tres causas:

- **Falta de luz o sombras intensas**: en imágenes con áreas oscuras, el modelo tiene dificultades para detectar el patrón rítmico de las hileras de viñedo. Esto se debe a que los contrastes que utiliza como pista para la segmentación se diluyen en condiciones de iluminación adversas. En estas regiones, las predicciones tienden a ser fragmentadas o ausentes.

- **Baja resolución efectiva**: aunque las imágenes tienen una resolución nominal suficiente, en algunos casos la densidad de píxeles por unidad de hilera es baja, especialmente cuando el viñedo aparece en ángulo o a distancia. Esto impide al modelo capturar detalles finos necesarios para trazar los contornos con precisión, resultando en segmentaciones borrosas o desplazadas.

- **Confusión por elementos visuales atípicos**: caminos, estructuras artificiales o cultivos vecinos con patrones lineales similares pueden inducir al modelo a falsas detecciones. Este fenómeno se observa sobre todo en zonas donde no se ha aplicado suficiente diversidad en el entrenamiento.

La combinación de estos factores explica el descenso de rendimiento en ciertos escenarios, y señala áreas clave donde intervenir con técnicas de mejora.

5.3. Análisis crítico de las máscaras de referencia

Las máscaras utilizadas como etiquetas fueron generadas de forma manual, y aunque se elaboraron cuidadosamente, presentan limitaciones propias del proceso humano, como errores en los bordes, zonas imprecisas o falta de homogeneidad. En múltiples casos, el modelo entrenado fue capaz de superar estas imperfecciones, produciendo segmentaciones más suaves, continuas y coherentes con la disposición real de las hileras de viñedo.

Este fenómeno introduce un sesgo en la evaluación: un valor de IoU bajo no siempre implica una predicción incorrecta, sino que puede deberse a imperfecciones en la máscara de referencia. Esta situación pone de manifiesto una limitación estructural en tareas supervisadas: la fiabilidad de la evaluación depende directamente de la calidad de las anotaciones.

5.4. Auto-refinamiento mediante entrenamiento incremental

Con el fin de explorar esta hipótesis y evaluar la capacidad del modelo para generar anotaciones más precisas, se diseñó una segunda fase de entrenamiento. En ella, se utilizó como conjunto de etiquetas las máscaras predichas por el primer modelo, en lugar de las máscaras manuales originales. Este enfoque permite transferir al nuevo modelo la capacidad de segmentación aprendida en la primera fase, pero con una supervisión más homogénea y ajustada a los patrones reales.

Este enfoque de auto-refinamiento, inspirado en técnicas de aprendizaje semi-supervisado, permitió al modelo beneficiarse de una supervisión más homogénea y alineada con los patrones visuales reales. Los resultados mostraron una mejora significativa en la métrica IoU en todos los conjuntos, lo que sugiere que el modelo puede automejorarse de forma incremental al liberarse de las limitaciones del etiquetado humano.

5.5. Resultados

Los principales hallazgos derivados de la fase de evaluación son los siguientes:

- El modelo ha demostrado una alta precisión en la segmentación, obteniendo un IoU de 0.8237 en validación.
- La calidad de las segmentaciones predichas supera en muchos casos a las máscaras manuales, especialmente en continuidad, alineación y precisión de los contornos.
- El valor de IoU no debe interpretarse de forma absoluta, sino en función de la calidad de las etiquetas utilizadas como referencia.
- La estrategia de autoetiquetado mediante redes previamente entrenadas se ha mostrado eficaz, permitiendo obtener métricas superiores en modelos posteriores.

- La validación cruzada geográfica ha revelado diferencias significativas en el rendimiento del modelo según el tipo de entorno, lo que subraya la importancia de considerar la heterogeneidad geográfica en la evaluación.
- Se han propuesto soluciones prácticas, como el uso de *data augmentation* avanzada o el auto-refinamiento, para superar las debilidades detectadas.

En conjunto, estos resultados consolidan la viabilidad del modelo como herramienta para aplicaciones agrícolas reales, y sugieren caminos claros para su mejora futura mediante técnicas accesibles y escalables.

5.6. Propuestas prácticas de mejora

A la luz de los resultados y del análisis crítico realizado, se identifican algunas propuestas concretas para abordar las debilidades observadas en la segmentación:

- ***Data augmentation* adicional**: aunque ya se emplearon técnicas básicas de aumento, podría incorporarse una mayor variedad de transformaciones (cambios de iluminación, condiciones atmosféricas sintéticas, deformaciones geométricas) para mejorar la robustez del modelo ante entornos complejos como los de la zona C.
- **Entrenamiento incremental iterativo**: repetir ciclos de autoetiquetado y reentrenamiento permite refinar progresivamente la segmentación, reduciendo la dependencia de anotaciones humanas y mejorando la adaptación del modelo a patrones reales.
- **Análisis de incertidumbre**: incorporar estimaciones de confianza en las predicciones ayudaría a identificar regiones con mayor probabilidad de error y a guiar procesos de etiquetado activo o revisión manual focalizada.
- **Fusión de modelos**: combinar diferentes variantes del modelo (entrenados con distintos hiperparámetros o resoluciones) mediante técnicas de *ensemble* podría mejorar la consistencia de las predicciones.
- **Revisión colaborativa de etiquetas**: utilizar herramientas de revisión visual asistida por IA podría mejorar significativamente la calidad de las máscaras, con un coste de etiquetado moderado.

Estas estrategias, combinadas, ofrecen un camino factible y eficaz para mejorar el rendimiento del sistema en escenarios difíciles sin necesidad de reetiquetar grandes volúmenes de datos.

Capítulo 6

Conclusiones y trabajo futuro

Este capítulo recoge las reflexiones finales derivadas del desarrollo del proyecto, tanto desde una perspectiva técnica como personal. En él se analizan los resultados alcanzados en relación con los objetivos planteados inicialmente, así como las lecciones aprendidas durante el proceso. Además, se plantean posibles líneas de mejora y evolución del sistema desarrollado, con una visión hacia su aplicabilidad real en el contexto agrícola.

6.1. Conclusiones

6.1.1. Perspectiva del proyecto

A lo largo del proyecto se han abordado y cumplido los objetivos establecidos en su fase inicial, logrando avanzar desde una etapa puramente investigadora hasta el desarrollo funcional de un sistema de segmentación de hileras de viñedo a partir de imágenes satelitales. A continuación se describe cómo se ha dado respuesta a cada uno de los objetivos:

- **OBJ-01: Investigación en modelos de IA para segmentación**
  Se ha llevado a cabo un estudio exhaustivo del estado del arte en técnicas de segmentación de imágenes, explorando múltiples arquitecturas relevantes como *FCN, DeepLab o ResUNet*. Finalmente, se optó por la arquitectura *U-Net* con *encoder ResNet34* preentrenado, al demostrar un rendimiento robusto, buena capacidad de generalización y una implementación eficiente para imágenes agrícolas.

- **OBJ-02: Adquisición y preprocesamiento de imágenes satelitales de zonas vitícolas**
  Para entrenar y validar el modelo se recopilaron imágenes de viñedos reales mediante la Sede Electrónica del Catastro español y *Google Maps*, centradas en zonas claramente delimitadas. Posteriormente, se aplicaron técnicas de preprocesamiento como redimensionamiento, normalización y binarización de máscaras, asegurando que tanto las imágenes como sus etiquetas tuvieran el formato necesario para el entrenamiento del modelo.

- **OBJ-03: Diseño y entrenamiento del modelo de segmentación *U-Net***
  Se desarrolló e implementó una red basada en la arquitectura *U-Net* con *encoder ResNet34*, entrenada con imágenes *RGB* de resolución 512×512. Se llevaron a cabo distintas pruebas de ajuste de hiperparámetros (tasa de aprendizaje, tamaño de *batch*, etc.), alcanzando una *IoU* de 0.8237 en validación, lo que indica un nivel alto de precisión en la segmentación de las hileras de viñedo.

- **OBJ-04: Posprocesamiento y análisis geométrico que permitan obtener métricas cuantificables**
  A partir de las máscaras generadas por el modelo es posible aplicar técnicas de análisis de imagen para extraer métricas agronómicas clave, como por ejemplo:

  - Longitud de las hileras (mediante detección de componentes conectados y trazado de líneas centrales).
  - Número total de hileras en cada parcela segmentada.
  - Densidad de plantación (relación entre número de hileras y superficie ocupada).
  - Orientación predominante de las hileras (a través de análisis de Fourier o de detección de líneas).

  Estas métricas permiten realizar análisis objetivos y cuantificables del terreno, aportando valor para aplicaciones en viticultura de precisión.

6.1.2. Perspectiva personal

Desde una perspectiva personal, este proyecto ha supuesto un reto multidisciplinar que ha implicado la integración de conocimientos en inteligencia artificial, procesamiento de imágenes, programación en Python y gestión de proyectos. Ha sido una experiencia de aprendizaje progresivo en la que se ha pasado de una etapa de exploración teórica a una implementación práctica completa y funcional.

A lo largo del trabajo, se han desarrollado competencias clave como la capacidad de análisis, el pensamiento crítico, la autonomía en la toma de decisiones técnicas, y la resolución de problemas reales con limitaciones computacionales y de datos. Además, se ha profundizado en el uso de herramientas profesionales como *PyTorch, GitHub, QGIS y LaTeX*, fortaleciendo habilidades valiosas para futuros entornos académicos o laborales.

El proyecto también ha sido una fuente de motivación, al comprobar cómo una solución basada en IA puede tener un impacto tangible en un contexto real como el sector agrícola, promoviendo la eficiencia, la automatización y la toma de decisiones basada en datos.

6.1.3. Implicaciones prácticas y posibles aplicaciones

El sistema desarrollado puede ser integrado en herramientas de apoyo a la gestión agronómica, permitiendo, por ejemplo:

- Automatizar el control de plantaciones en explotaciones vitícolas.
- Detectar irregularidades o deficiencias en la distribución de hileras.
- Complementar análisis multitemporales para evaluar cambios en el viñedo.

Además, esta solución abre la puerta a posibles vías de monetización o transferencia tecnológica, como el desarrollo de un servicio *SaaS (Software as a Service)* accesible para cooperativas agrícolas, bodegas o empresas de teledetección, que permita cargar imágenes y obtener análisis segmentados directamente desde la web o mediante *API*. A partir de la longitud de las filas de viñedo es posible estimar el número total de cepas, siempre que se conozca la distancia promedio ocupada por cada una. Con esta información, y considerando la productividad media de una cepa, puede calcularse una estimación aproximada del rendimiento total del viñedo. Además, si la uva se destina a la elaboración de vino u otras bebidas, también se puede estimar la cantidad de producto final obtenido, así como su valor económico, conociendo el porcentaje de uva aprovechable y el precio por litro del vino.

6.2. Trabajo futuro

A pesar de los buenos resultados obtenidos, el proyecto deja abiertas múltiples vías de mejora y evolución. A continuación se detallan algunas líneas de trabajo futuras con especial atención a su viabilidad técnica y científica:

- **Ampliación del conjunto de datos**: incluir imágenes de distintas regiones vitícolas, diferentes estaciones del año y condiciones de cultivo, así como diversas resoluciones (tanto mayores como menores), para mejorar la capacidad de generalización y robustez del modelo frente a escenarios heterogéneos.
- **Transferencia geográfica y adaptación a otros cultivos**: sería relevante estudiar la capacidad del modelo para generalizar más allá de su dominio original. Esto incluye evaluar su rendimiento en otras zonas geográficas con patrones distintos de plantación, así como su aplicación a cultivos similares estructuralmente, como olivos, almendros u hileras de frutales, mediante técnicas de *transfer learning* o adaptación de dominio.
- **Uso de imágenes multiespectrales o infrarrojas**: la incorporación de bandas más allá del espectro visible permitiría mejorar la detección de vegetación activa y la delimitación precisa de las hileras, especialmente en momentos en los que las diferencias visuales no son tan marcadas (por ejemplo, en invierno o en cultivos jóvenes).
- **Refinamiento automático de etiquetas y aprendizaje semisupervisado**: se podría experimentar con técnicas de *self-training* o *pseudo-labeling* que permitan ampliar el conjunto de entrenamiento sin necesidad de etiquetado manual, utilizando como punto de partida las propias predicciones del modelo o una red auxiliar.

- **Mejora de la aplicabilidad técnica del sistema**: aún quedan aspectos técnicos por resolver para facilitar la adopción real de esta tecnología en entornos productivos, como:
  - Optimización del modelo para ejecutarse en dispositivos de bajo coste o entornos de *edge computing* (por ejemplo, embarcado en drones o estaciones remotas).
  - Automatización completa del *pipeline*, desde la entrada de imágenes (satélite o dron) hasta la obtención de métricas clave como la longitud de hileras o su orientación.
  - Robustez ante condiciones no ideales como imágenes parcialmente nubladas, rotaciones, inclinaciones, ruido satelital o cambios de escala.
- **Estudios adicionales sobre escalabilidad**: sería recomendable evaluar cómo se comporta el sistema al aplicarlo a grandes extensiones de terreno (por ejemplo, a nivel regional), integrando mosaicos satelitales completos, y analizando los tiempos de procesamiento, consumo de recursos y costes asociados.
- **Integración con plataformas digitales del sector agrícola**: se propone explorar la compatibilidad e integración del modelo en sistemas de información geográfica (SIG), plataformas de gestión agronómica o servicios de teledetección ya existentes (por ejemplo, *Agroptima*, *EOS Crop Monitoring* o herramientas basadas en *QGIS/ArcGIS*), con el fin de facilitar su adopción por parte de usuarios no técnicos y maximizar su valor práctico.
- **Estudios de transferencia tecnológica**: como evolución natural del presente trabajo, se sugiere investigar la viabilidad de trasladar este prototipo de investigación a un producto comercial o servicio tecnológico. Esto podría incluir colaboraciones con empresas del sector agroalimentario, *startups* de *agrotech* o instituciones públicas interesadas en la monitorización territorial y la agricultura de precisión.

Estas líneas de trabajo permitirían que el sistema evolucionase desde un prototipo académico hacia una solución práctica, escalable y transferible, con potencial real de impacto en la viticultura de precisión y, más ampliamente, en el ámbito de la agricultura inteligente.

Parte III

Apéndices

Apéndice A

Manual de Instalación

El contenido se presenta en forma de *scripts* de Python con extensión `.py`. Para poder ejecutarlos, es necesario realizar algunas comprobaciones e instalaciones previas. A continuación, se detalla el proceso para su ejecución en un sistema operativo Linux:

1. Descargar Python y herramientas útiles.
   `sudo apt install python3 python3-pip python3-venv`
2. Crear y activar un entorno virtual para gestionar las dependencias de forma independiente.
   a) Crear el entorno virtual
   `python3 -m venv nombre_del_entorno`
   b) Moverse con el comando `cd` hasta el directorio donde se desee crear el entorno.
   c) Activar el entorno virtual
   `source nombre_del_entorno/bin/activate`
3. Instalar las librerías en las versiones necesarias:
   a) `torch==2.6.0`
   b) `torchvision==0.21.0`
   c) `matplotlib==3.10.1`
   d) `opencv-python==4.11.0.86`
   e) `numpy==1.26.4`
   f) `pandas==2.2.3`
   g) `scikit-learn==1.6.1`
   h) `tqdm==4.67.1`
   i) `Pillow==11.1.0`

Al finalizar para desconectar el entorno se emplea el comando `deactivate`.

Apéndice B

Contenido adjunto

El proyecto se compone de cuatro *scripts* principales escritos en Python:

- `train.py`: implementa el proceso de entrenamiento del modelo de segmentación.
- `train2.py`: implementa el proceso de entrenamiento del modelo de segmentación con validación cruzada geográfica.
- `predict.py`: permite realizar inferencias sobre nuevas imágenes, generando las máscaras correspondientes.
- `process.py`: se encarga de realizar la inferencia de imágenes y analizar las máscaras generadas para extraer información agronómica relevante, la cual se guarda en archivos con formato `JSON`.
- `processone.py`: se encarga de realizar la inferencia de una imagen concreta y analizar la máscara generada para extraer información agronómica relevante, la cual se guarda en archivos con formato `JSON`.

La estructura del directorio destinada al almacenamiento de imágenes, máscaras y datos de test (junto con sus respectivos archivos `.txt`, que contienen la escala de cada imagen) es la siguiente:

```
dataset/
    images/
    masks/
    test/
```

Para la validación cruzada geográfica la estructura es la siguiente:

```
dataset/
    images/
        A/
```

```
    B/
    C/
masks/
test/
```

Además, el repositorio incluye:

- El modelo entrenado en formato `vinedos_model.pth`.
- La gráfica de evolución del entrenamiento.
- Los modelos entrenados mediante validación cruzada `model_fold_X.pth`.
- Las gráficas de evolución del entrenamiento de los modelos entrenados mediante validación cruzada.
- La gráfica de evolución del entrenamiento.
- Un archivo `requirements.txt` con las librerías y versiones necesarias.
- Carpetas que contienen las máscaras generadas por el modelo y los datos extraídos a partir de ellas.
- Un archivo `dataset_meta.csv` que contiene metadatos asociados a cada imagen del conjunto de datos. Cada fila representa una muestra (una imagen y su máscara correspondiente), e incluye columnas clave como el nombre del archivo (filename) y la zona geográfica a la que pertenece (zone).
- Un archivo `README` con la información necesaria para comprender el funcionamiento y uso del proyecto.

Bibliografía

[1] *A complete guide to classification metrics in machine learning*. Evidently AI. URL: https://www.evidentlyai.com/classification-metrics.

[2] Hamed Habibi Aghdam y Elnaz Jahani Heravi. *Guide to Convolutional Neural Networks*. Springer International Publishing, 2017.

[3] *AgroSpace*. AgroSpace. URL: https://agrospace.cl/?trk=public_post_comment-text.

[4] Ayamga et al. "Women empowerment and food-nutrition security in Sierra Leone: The Gender Model Family approach". En: *Nature* 12.1 (2023).

[5] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". En: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), págs. 580-587.

[6] H.G. Barrow y J.M. Tenenbaum. "Computational vision". En: *Proceedings of the IEEE* 69.5 (1981), págs. 572-595.

[7] *Bases y tipos de cotización*. Seguridad Social. URL: https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537.

[8] Anibal Bregón Bregón. *Apuntes tema 4 Sistemas Inteligentes*. 2024-2025.

[9] Anibal Bregón Bregón. *Apuntes tema 7 Sistemas Inteligentes*. 2024-2025.

[10] Anibal Bregón Bregón. *Apuntes tema 8 Sistemas Inteligentes*. 2024-2025.

[11] L. Breiman. "Random Forests". En: *Machine Learning* 45 (2001), págs. 5-32.

[12] A. Casado-García et al. "Generalization of deep learning models applied to semantic segmentation of in-field natural images in vineyards". En: *Precision agriculture '23* (2023), 385–392.

[13] *ChatGPT*. OpenAI. URL: https://www.openai.com/chatgpt.

[14] Agnese Chiatti et al. *Surgical fine-tuning for Grape Bunch Segmentation under Visual Domain Shifts*. arXiv, 2023.

[15] *Crear Red Neuronal desde las matemáticas*. European Valley. URL: https://www.europeanvalley.es/noticias/crear-red-neuronal-desde-las-matematicas/.

[16] *DeepWeeds*. GitHub. URL: https://github.com/AlexOlsen/DeepWeeds.

[17] *El Método PERT*. Enredando Proyectos. URL: https://enredandoproyectos.com/el-metodo-pert/.

[18] *El perceptrón como neurona artificial*. Jose Mariano Alvarez. URL: https://blog.josemarianoalvarez.com/2018/06/10/el-perceptron-como-neurona-artificial/.

[19] Gallo et al. "Enhancing Crop Segmentation in Satellite Image Time Series with Transformer Networks". En: *Remote Sensing* 16.12 (2024), pág. 2341. URL: https://www.researchgate.net/publication/379546635_Enhancing_crop_segmentation_in_satellite_image_time-series_with_transformer_networks.

[20] *GitHub*. GitHub. URL: https://github.com.

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

[22] Kaiming He et al. "Deep Residual Learning for Image Recognition". En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, págs. 770-778.

[23] P.M. Institute. *Guide to the Project Management Body of Knowledge (PMBOK Guide). Sixth Edition*. Project Management Institute, 2017.

[24] *Inteligencia artificial*. Wikipedia. URL: https://es.wikipedia.org/wiki/Inteligencia_artificial.

[25] *Intersección sobre Unión (IoU)*. Ultralytics. URL: https://www.ultralytics.com/es/glossary/intersection-over-union-iou.

[26] Paul Jaccard. "The Distribution of the Flora in the Alpine Zone". En: *New Phytologist* 11.2 (1912), págs. 37-50.

[27] Teja Kattenborn et al. "Review on Convolutional Neural Networks (CNN) in vegetation remote sensing". En: *ISPRS Journal of Photogrammetry and Remote Sensing* 173 (2021), págs. 24-49.

[28] Diederik P. Kingma y Jimmy Ba. "Adam: A Method for Stochastic Optimization". En: *International Conference on Learning Representations (ICLR)* (2015).

[29] Irina Korotkova y Natalia Efremova. *AI for Agriculture: the Comparison of Semantic Segmentation Methods for Crop Mapping with Sentinel-2 Imagery*. arXiv, 2023.

[30] *LaTeX: A Document Preparation System*. Leslie Lamport. URL: https://www.latex-project.org/.

[31] *Leading Data Annotation Platform*. CVAT. URL: https://www.cvat.ai/.

[32] Yann Lecun, Yoshua Bengio y Geoffrey Hinton. "Deep learning". En: *HAL Open Science* 521.7553 (2023), págs. 436-444.

[33] MacDonald et al. "VistaFormer: Scalable Vision Transformers for Satellite Image Time Series Segmentation". En: *arXiv preprint arXiv:2409.08461* (2024). URL: https://arxiv.org/abs/2409.08461.

[34] *Mendeley - Reference Management Software*. Mendeley Ltd. URL: https://www.mendeley.com/.

[35] *Microsoft Teams*. Microsoft Corporation. URL: https://www.microsoft.com/es-es/microsoft-teams/group-chat-software.

[36] *Métricas*. IBM. URL: https://www.ibm.com/docs/es/masv-and-l/maximo-vi/cd?topic=configuring-understanding-metrics.

[37] *Overleaf: Online LaTeX Editor*. Overleaf. URL: https://www.overleaf.com.

[38] *Proyecto Jupyter*. Wikipedia. URL: https://es.wikipedia.org/wiki/Proyecto_Jupyter.

[39] Manav Raj y Robert Seamans. "Primer on artificial intelligence and robotics". En: *Journal of Organization Design* 8.11 (2019).

[40] Laukik Raut, Rajat Wakode y Pravin Talmale. "Overview on Kanban Methodology and its Implementation". En: *International Journal for Scientific Research Development* 03 (2015), págs. 2518-2521.

[41] Olaf Ronneberger, Philipp Fischer y Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". En: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)* 9351 (2015).

[42] David E Rumelhart, Geoffrey E Hinton y Ronald J Williams. "Learning representations by back-propagating errors". En: *Nature* 323.6088 (1986), págs. 533-536.

[43] *Segmentación*. Juan Sensio. URL: https://www.juansensio.com/blog/050_cv_segmentacion.

[44] *Segmentación semántica Tres cosas que es necesario saber*. MathWorks. URL: https://es.mathworks.com/solutions/image-video-processing/semantic-segmentation.html.

[45] *Sentinel-2 for Agriculture*. Copernicus. URL: https://www.copernicus.eu/en/sentinel-2-agriculture.

[46] Shai Shalev-Shwartz y Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[47] *Spatial without Compromise*. QGIS. URL: https://qgis.org/.

[48] Girma Tariku et al. "Semantic Segmentation of Land Covers Using Deep Learning with a Pre-trained Backbone: A Case Study in in the Franciacorta Wine-growing Area". En: *Preprints* (2024).

[49] T.Barros et al. "Multispectral vineyard segmentation: A deep learning comparison study". En: *Computers and Electronics in Agriculture* 195 (2022).

[50] *The 2020 Scrum Guide*. ScrumGuides. URL: https://scrumguides.org/scrumguide.html.

[51] Jorge Torres-Sánchez et al. "Multi-temporal mapping of the vegetation fraction in early-season wheat fields using images from UAV". En: *Computers and Electronics in Agriculture* 103 (2015), 104–113.

[52] A. M. Turing. "Computing Machinery and Intelligence". En: *Mind* 59.236 (1950), págs. 433-460.

[53] *U-Net*. Wikipedia. URL: https://es.wikipedia.org/wiki/U-Net.

[54] *U-NET : todo lo que tienes que saber sobre la red neuronal de Computer Vision*. Data Scientest. URL: https://datascientest.com/es/u-net-lo-que-tienes-que-saber.

[55] *U-NET para segmentación semántica, explicación del paper*. Pepe Cantoral, Ph.D. URL: https://www.youtube.com/watch?v=waIPUsecaaQ&ab_channel=PepeCantoral%2CPh.D..

[56] *VineView*. VineView. URL: https://vineview.es/.

[57] *Visión Artificial - Implementando UNet desde cero (parte 1)*. Juan Sensio. URL: https://www.youtube.com/watch?v=5Qm2TQRtn9A&ab_channel=SensIO.

[58] *Visión Artificial - Implementando UNet desde cero (parte 2)*. Juan Sensio. URL: https://www.youtube.com/watch?v=PCK6K1ei34c&ab_channel=SensIO.

[59] *Visión Artificial - Segmentación Semántica*. Juan Sensio. URL: https://www.youtube.com/watch?v=NZcMZTpR_j8&ab_channel=SensIO.

[60] *Visual Studio Code*. Microsoft Corporation. URL: https://code.visualstudio.com/.

[61] Yueyong Wang et al. "Sh-DeepLabv3+: An Improved Semantic Segmentation Lightweight Network for Corn Straw Cover Form Plot Classification". En: *Agriculture* 14 (2024).

[62] Samir S. Yadav y S. Jadhav. "Deep convolutional neural network based medical image classification for disease diagnosis". En: *Journal of Big Data* 6 (2019).

[63] Y.Lecunetal. "Gradient-based learning applied to document recognition". En: *Proceedings of the IEEE* 86.11 (1998), págs. 2278-2324.

[64] *¿Qué es el aprendizaje automático?* IBM. URL: https://www.ibm.com/mx-es/think/topics/machine-learning.

[65] *¿Qué es el aprendizaje no supervisado?* IBM. URL: https://www.ibm.com/es-es/think/topics/unsupervised-learning.

[66] *¿Qué es el aprendizaje por transferencia?* IBM. URL: https://www.ibm.com/es-es/think/topics/transfer-learning.

[67] *¿Qué es el aprendizaje supervisado?* IBM. URL: https://www.ibm.com/es-es/think/topics/supervised-learning.

[68] *¿Qué es la inteligencia artificial (IA)?* IBM. URL: https://www.ibm.com/es-es/think/topics/artificial-intelligence.

[69] *¿Qué es una GAN?* Amazon. URL: https://aws.amazon.com/es/what-is/gan/.

[70] *¿Qué son las redes neuronales?* IBM. URL: https://www.ibm.com/es-es/think/topics/neural-networks.