GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DETECCIÓN AUTOMÁTICA DE DEFECTOS EN TÚNELES FERROVIARIOS

Autor: Antía García Rivero

Director: Luis Francisco Sánchez Merchante

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Detección automática de defectos en túneles ferroviarios

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2024/2025 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Antía García Rivero Fecha: …25…/ …08…/ …2025…

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

2025.08.26 13:02:46 +02'00'

Fdo.: Luis Francisco Sánchez Merchante Fecha: 26/08/2025

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DETECCIÓN AUTOMÁTICA DE DEFECTOS EN TÚNELES FERROVIARIOS

Autor: Antía García Rivero

Director: Luis Francisco Sánchez Merchante

Madrid

Agradecimientos

A mi familia, por su constante sacrificio y cariño y por darme la mano en cada momento importante de mi vida.

A mi tutor, por su paciencia, implicación y atención durante todo este proceso.

A mi amigo Nacho, por ser mi familia fuera de casa y apoyarme como tal.

A Álvaro, por ser el mejor animador que se puede pedir.

A David Gistau y a Axil Ingeniería y Servicios, porque sin ellos el desarrollo de este trabajo no hubiera sido posible.

DETECCIÓN AUTOMÁTICA DE DEFECTOS EN TÚNELES FERROVIARIOS

Autor: García Rivero, Antía.
Director: Sánchez Merchante, Luis Francisco.
Entidad Colaboradora: Axil Ingeniería y Servicios

RESUMEN DEL PROYECTO

Se ha desarrollado un sistema de detección automática de defectos en túneles ferroviarios mediante el uso de la arquitectura U-Net y una de sus variantes, la Attention U-Net. Se han realizado varias pruebas sobre conjuntos de imágenes recortadas y etiquetadas con distintos defectos. Los resultados muestran que, con conjuntos balanceados la red logra segmentaciones estables y coherentes, mientras que en escenarios desbalanceados, la Attention U-Net mejora el rendimiento frente a la U-Net clásica.

Palabras clave: Segmentación de imágenes, U-Net, Attention U-Net, túneles ferroviarios, detección de defectos, aprendizaje profundo

1. Introducción

Para garantizar la seguridad y durabilidad de los túneles ferroviarios, la inspección de dicha estructura es una tarea crítica. Estos espacios presentan condiciones particularmente complejas para la supervisión, al ser entornos cerrados, con baja iluminación natural y sometidos a tensiones estructurales constantes. Hoy en día, una parte mayoritaria de este proceso es realizada de forma manual, mediante recorridos físicos o revisión visual de miles de imágenes. Dicho enfoque, aunque efectivo, implica una fuerte inversión de tiempo y recursos, además de ser sujeto de exposición a errores humanos (Samsami, 2024).

En este contexto, las técnicas de *deep learning* basadas en segmentación ofrecen una alternativa prometedora para automatizar y optimizar la detección de defectos. Ampliamente validada en segmentación biomédica, la arquitectura U-Net (Ronneberger et al., 2015)ofrece un marco adecuado para tareas de segmentación semántica en imágenes complejas, pudiendo identificar defectos con precisión a nivel de píxel. En colaboración con la empresa Axil Ingeniería y Servicios, el presente trabajo explora su viabilidad para el caso específico de túneles ferroviarios.

2. Definición del proyecto

El objetivo de este Trabajo de Fin de Grado es validar el uso de la arquitectura U-Net e la detección automática de defectos en túneles ferroviarios a partir de imágenes proporcionadas por la empresa colaboradora. El trabajo se centró en dos patologías representativas: grafitis (rosa) y humedades (azul). Los objetivos específicos fueron:

- Entrenar y evaluar el modelo en defectos representativos
- Creación de un *dataset* de imágenes de túneles etiquetadas para realizar investigación sobre la detección de defectos en túneles
- Valorar la aplicabilidad del sistema en los procesos de inspección de la empresa

3. Descripción del modelo/sistema/herramienta

El flujo de trabajo se organizó en dos bloques principales:

1. Procesamiento de datos: utilizando como base los ficheros de la empresa con las patologías anotadas de color, se generaron máscaras binarias para cada defecto. Posteriormente, tanto las imágenes como las máscaras se recortaron en bloques de 256x256 píxeles con un 50% de solapamiento, asegurando que los defectos pequeños quedasen representados íntegramente en al menos un recorte.
2. Modelado: se implementó la arquitectura U-Net clásica en TensorFlow/Keras. También se realizaron pruebas con Attention U-Net, introducida por Oktay et al. (2018). Se evaluaron diferentes funciones de pérdida como *Dice Loss, Binary Cross Entropy y Tversky Loss*. La función de pérdida que proporcionó mejores resultados fue la combinación de *Binary Crossentropy* y *Dice Loss*, equilibrando estabilidad y sensibilidad a clases minoritarias.

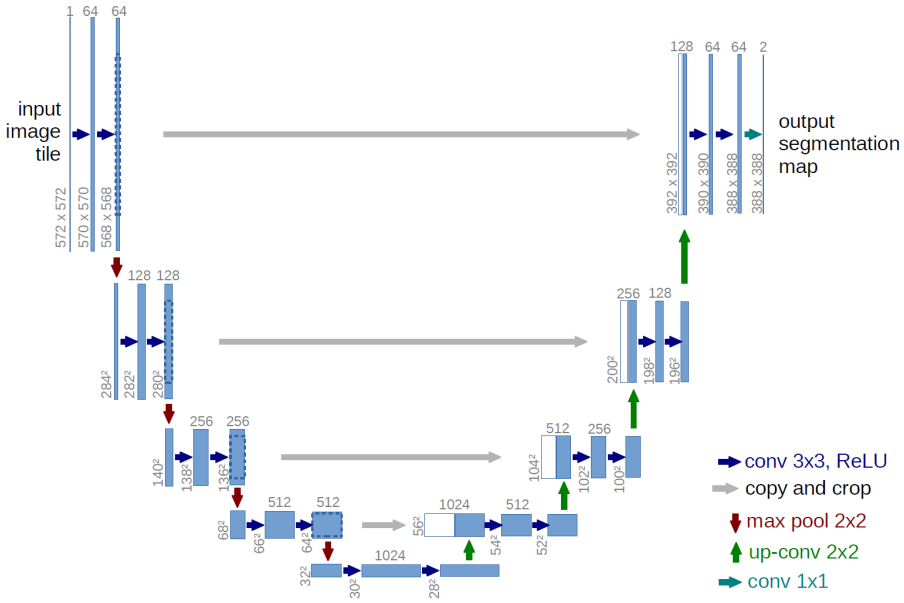


Figura 1. Arquitectura de la U-Net original. Fuente: Ronneberger, O., Fischer, P., & Brox, T. (2015)

4. Resultados

Los resultados obtenidos ponen de manifiesto la viabilidad del uso de U-Net en la detección automática de defectos

- Grafitis: con un conjunto de 1547 imágenes balanceado con 80% con defecto, alcanzó un coeficiente Dice de 0.8408 en entrenamiento, 0.7382 en validación y 0.8290 en *test*. Estos resultados demuestran que, aún con un conjunto de datos reducido, el modelo fue capaz de aprender patrones representativos y generalizar de forma razonable.
- Humedades:
    - Conjunto desbalanceado: en un experimento con 6.000 imágenes elegidas aleatoriamente, donde predominaban los casos sin defecto, se probaron dos arquitecturas:

- U-Net clásica: el rendimiento disminuyó drásticamente, con coeficientes Dice de 0.3594 en entrenamiento, 0.3312 en validación y 0.6726 en *test*. El impacto del desbalance de clases coincide con lo señalado en estudios previos sobre segmentación con conjuntos de datos limitados, donde la proporción de ejemplos positivos condiciona fuertemente el aprendizaje (Terven et al., 2025).
- Attention U-Net: sin utilizar un conjunto de validación para evitar un *early stopping* prematuro, se alcanzó un coeficiente Dice de 0.8530 en entrenamiento y 0.7673 en *test*.

o Conjunto balanceado: al utilizar un conjunto de 5.248 imágenes con un 80% de imágenes con defecto, el rendimiento fue significativamente superior, alcanzando un coeficiente Dice de 0.9785 en entrenamiento, 0.9400 en validación y 0.8900 en test. Se observó que disponer de un mayor número de ejemplos con defecto permitió un aprendizaje más robusto y estable. En la imagen incluida a continuación se pueden observar las predicciones realizadas con dicho modelo:

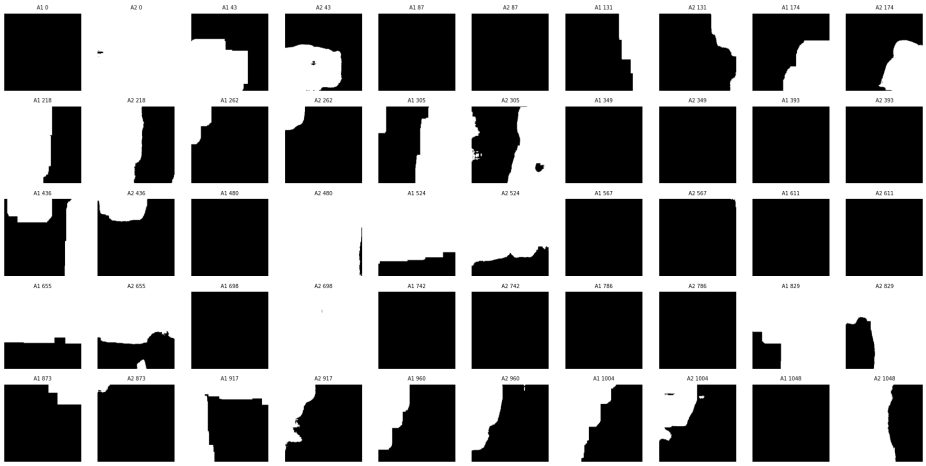


Figura 2. Comparación de máscaras reales y predicciones del modelo en el conjunto de prueba

5. Conclusiones

El trabajo constituye una prueba de concepto sólida, validando el uso de la U-Net como manera de automatizar parcialmente la segmentación de defectos en túneles ferroviarios, reduciendo la carga de trabajo manual y dirigiendo la atención de los técnicos hacia aquellas regiones con mayor probabilidad de presentar patologías. La robustez de la U-Net en este contexto valida su potencial de aplicarse a entornos reales de inspección industrial, en línea con investigaciones recientes sobre grafitis en estructuras (Fogaça et al., 2023) .

6. Referencias

Fogaça, J., Brandão, T., & Ferreira, J. C. (2023). Deep Learning-Based Graffiti Detection: A Study Using Images from the Streets of Lisbon. *Applied Sciences (Switzerland)*, *13*(4). https://doi.org/10.3390/app13042249

Oktay, O., Schlemper, J., Folgoc, L. Le, Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N. Y., Kainz, B., Glocker, B., & Rueckert, D. (2018). *Attention U-Net: Learning Where to Look for the Pancreas*. http://arxiv.org/abs/1804.03999

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9351*, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Samsami, R. (2024). A Systematic Review of Automated Construction Inspection and Progress Monitoring (ACIPM): Applications, Challenges, and Future Directions. In *CivilEng* (Vol. 5, Issue 1, pp. 265–287). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/civileng5010014

Terven, J., Cordova-Esparza, D. M., Romero-González, J. A., Ramírez-Pedraza, A., & Chávez-Urbiola, E. A. (2025). A comprehensive survey of loss functions and metrics in deep learning. *Artificial Intelligence Review*, *58*(7). https://doi.org/10.1007/s10462-025-11198-7

AUTOMATIC DETECTION OF DEFECTS IN RAILWAY TUNNELS

Author: García Rivero, Antía.
Supervisor: Sánchez Merchante, Luis Francisco.
Collaborating Entity: Axil Ingeniería y Servicios

ABSTRACT

This work presents a system for the automatic detection of defects in railway tunnels, based on the U-Net architecture and one of its variants, the Attention U-Net. The approach was validated using cropped and labeled image datasets containing different types of defects. Experimental results indicate that, in balanced datasets, both architectures achieve consistent and accurate segmentations. However, in imbalanced scenarios, the Attention U-Net demonstrates superior performance compared to the classical U-Net.

Keywords: Image segmentation, U-Net, Attention U-Net, railway tunnels, defect detection, deep learning

1. Introduction

Ensuring the safety and durability of railway tunnels makes their inspection a critical task. These structures present particularly complex conditions for supervision, as they are enclosed environments, with low natural lighting and subject to constant structural stress. Currently, a large part of this process is still carried out manually, either through physical inspections or by visually reviewing thousands of images. Although effective, this approach requires a significant investment of time and resources and is subject to human error (Samsami, 2024).

In this context, deep learning techniques based on segmentation offer a promising alternative to automate and optimize defect detection. Widely validated in biomedical segmentation, the U-Net architecture (Ronneberger et al., 2015) provides a suitable framework for semantic segmentation tasks in complex images, enabling pixel-level defect identification. In collaboration with the company Axil Ingeniería y Servicios, this work explores its feasibility in the specific case of railway tunnels.

2. Project Definition

The aim of this Bachelor's Thesis is to validate the use of the U-Net architecture for the automatic detection of defects in railway tunnels from images provided by the collaborating company. The work focused on two representative pathologies: graffiti (pink) and humidity (blue). The specific objectives were:

- To train and evaluate the model on representative defects
- To create a dataset of labeled tunnel images for defect detection research
- To assess the applicability of the system in the company's inspection process

3. Descripción del modelo/sistema/herramienta

The workflow was organized into two main blocks:

1. Data processing: Based on the company's files with color-annotated defects, binary masks were generated for each defect. Subsequently, both the images and the masks were cropped into 256×256-pixel patches with 50% overlap, ensuring that small defects were fully represented in at least one crop.

2. Model: The classical U-Net architecture was implemented in TensorFlow/Keras. Additional experiments were conducted with the Attention U-Net, introduced by Oktay et al. (2018). Different loss functions were evaluated: *Dice Loss, Binary Crossentropy and Tversky Loss*. The loss function that provided the best training was a combination of Binary Crossentropy and Dice Loss, balancing stability and sensitivity to minority classes.

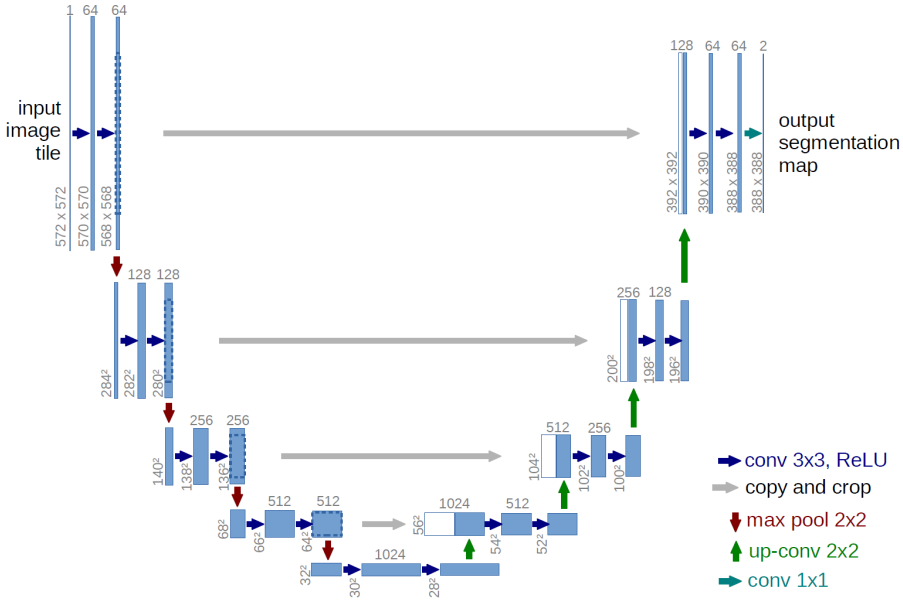


Figure 1. Original U-Net architecture. Source: Ronneberger, O., Fischer, P., & Brox, T. (2015)

4. Results

The obtained results highlight the feasibility of using U-Net for the automatic detection of defects.

- Graffiti: with a dataset of 1,547 images balanced with 80% containing defects, the model achieved a Dice coefficient of 0.8408 in training, 0.7382 in validation, and 0.8290 in testing. These results demonstrate that, even with a relatively small dataset, the model was able to learn representative patterns and generalize reasonably well.

- Humidity:

    - Imbalanced dataset: in an experiment with 6,000 randomly selected images, where non-defect cases predominated, two architectures were tested:

- *Classical U-Net:* performance decreased drastically, with Dice coefficients of 0.3594 in training, 0.3312 in validation, and 0.6726 in testing. The impact of class imbalance is consistent with previous studies on segmentation with limited datasets, where the proportion of positive examples strongly conditions learning (Terven et al., 2025).

- *Attention U-Net:* without using a validation set, to avoid premature early stopping, the model achieved Dice coefficients of 0.8530 in training and 0.7673 in testing.

- Balanced dataset: using a dataset of 5,248 images with 80% containing defects, performance was significantly higher, reaching a Dice coefficient of 0.9785 in training, 0.9400 in validation, and 0.8900 in testing. It was observed that having a larger number of defect examples enabled more robust and stable learning. In the figure below, representative predictions obtained with this model can be observed:

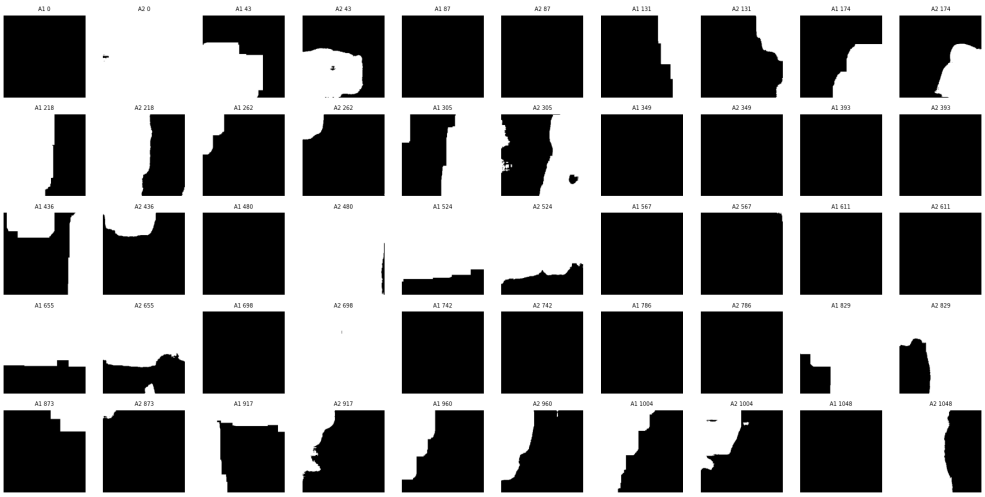


Figure 2. Comparison of ground-truth masks and model predictions on the test set

5. Conclusions

This work constitutes a solid proof of concept, validating the use of U-Net as a way to partially automate the segmentation of defects in railway tunnels, thereby reducing manual workload and directing technicians' attention to regions with a higher probability of presenting pathologies. The robustness of U-Net in this context supports its potential application in real industrial inspection environments, in line with recent research on graffiti in structures (Fogaça et al., 2023).

6. References

Fogaça, J., Brandão, T., & Ferreira, J. C. (2023). Deep Learning-Based Graffiti Detection: A Study Using Images from the Streets of Lisbon. *Applied Sciences (Switzerland)*, *13*(4). https://doi.org/10.3390/app13042249

Oktay, O., Schlemper, J., Folgoc, L. Le, Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N. Y., Kainz, B., Glocker, B., & Rueckert, D. (2018). *Attention U-Net: Learning Where to Look for the Pancreas*. http://arxiv.org/abs/1804.03999

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9351*, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Samsami, R. (2024). A Systematic Review of Automated Construction Inspection and Progress Monitoring (ACIPM): Applications, Challenges, and Future Directions. In *CivilEng* (Vol. 5, Issue 1, pp. 265–287). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/civileng5010014

Terven, J., Cordova-Esparza, D. M., Romero-González, J. A., Ramírez-Pedraza, A., & Chávez-Urbiola, E. A. (2025). A comprehensive survey of loss functions and metrics in deep learning. *Artificial Intelligence Review*, *58*(7). https://doi.org/10.1007/s10462-025-11198-7

Índice de la memoria

Capítulo 1. Introducción 20

Capítulo 2. Estado de la Cuestión 24

Capítulo 3. Fundamentos teóricos 32

3.1 Introducción a la segmentación de imágenes 32

3.2 Arquitectura U-Net 34

3.2.1 Flujo de datos 36

3.3 Entrenamiento de una U-Net 38

3.4 Tipos de U-Net 40

Capítulo 4. Descripción de tecnologías 45

4.1 Librerías principales 45

4.2 Recursos Hardware 46

Capítulo 5. Trabajo desarrollado 47

5.1 Procesamiento de datos 47

5.1.1 Creación de las máscaras 48

5.1.2 Recorte en 256x256 con solapamiento 52

5.1.3 Data Augmentation 54

5.2 Entrenamiento de la red U-Net 55

Capítulo 6. Resultados y discusión 58

6.1 Grafitis 58

6.2 Humedades 61

6.2.1 Conjunto desbalanceado 61

6.2.2 Conjunto balanceado 63

Conclusiones y Trabajos Futuros 68

6.3 Estimación del volumen de datos 69

6.4 Trabajos futuros 70

Capítulo 7. Bibliografía ***72***

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS ***76***

ANEXO II: Función solidificación máscaras ***77***

ANEXO III: Generación máscara azul ***78***

ANEXO IV: Generación máscara rosa ***79***

ANEXO V: Generación máscara amarilla ***80***

Índice de figuras

Figura 1. Arquitectura de la U-Net original. Fuente: Ronneberger, O., Fischer, P., & Brox, T. (2015) 8
Figura 3. Arquitectura de la U-Net++. Fuente: Stoyanov et al., 2018. 42
Figura 4. Arquitectura de la ResUNet. Fuente: Zhang et al., 2018. 43
Figura 5. Leyenda de patologías. 48
Figura 6. Ejemplo de imagen de ficheros PAT con patologías rosas 50
Figura 7. Máscara rosa correspondiente a la Figura 6. 50
Figura 8. Imagen de ficheros PAT con patologías azules 51
Figura 9. Máscara azul correspondiente a la Figura 8 51
Figura 10. Coeficiente Dice de los conjuntos de entrenamiento y validación a lo largo de las épocas 59
Figura 11. Función de pérdida Dice de los conjuntos de entrenamiento y validación a lo largo de las épocas 59
Figura 12. Comparación de máscaras reales y predicciones del modelo en el conjunto de prueba 60
Figura 13. Coeficiente Dice de los conjuntos de entrenamiento y validación a lo largo de las épocas 62
Figura 14. Función de pérdida Dice de los conjuntos de entrenamiento y validación a lo largo de las épocas 62
Figura 15. Comparación de máscaras reales y predicciones del modelo en el conjunto de prueba 63
Figura 16. Coeficiente Dice de los conjuntos de entrenamiento y validación a lo largo de las épocas 65
Figura 17. Función de pérdida Dice de los conjuntos de entrenamiento y validación a lo largo de las épocas 65

Figura 18. Comparación de máscaras reales y predicciones del modelo en el conjunto de prueba 66

Índice de tablas

Tabla 1. Comparación de los estudios revisados 31
Tabla 2. Comparación de los resultados obtenidos 67

Capítulo 1. INTRODUCCIÓN

La capacidad para desplazar personas y mercancías ha determinado la expansión de ciudades, el comercio internacional, las relaciones sociales y el acceso a servicios fundamentales. Es por ello por lo que el transporte puede considerarse como eje estructurador de las sociedades modernas. En concreto, el transporte terrestre ha tenido un protagonismo destacado en la configuración de territorio, ejerciendo un impacto especialmente profundo y duradero en términos tanto espaciales como económicos gracias al desarrollo de carreteras, tranvías y ferrocarriles que han facilitado la movilidad interregional y la interconexión entre zonas urbanas y rurales.

Durante la revolución industrial, ante una creciente necesidad de transporte de materias primas, bienes manufacturados y personas, surge el ferrocarril en Europa. Rápidamente se volvió un instrumento clave para la expansión industrial y urbana del siglo XIX, actuando como catalizador del crecimiento urbano, conectando centros productivos, promoviendo nuevas urbanizaciones y redefiniendo la jerarquía espacial de los territorios. Desde hace décadas, las políticas de ordenación territorial en Europa han integrado el ferrocarril como eje clave para promover ciudades más compactas, eficientes y sostenibles, contrastando con los modelos dispersos que tienden a surgir en torno al automóvil privado (Alpkokin, 2012).

Son múltiples los estudios que tratan la influencia del ferrocarril en los procesos de urbanización. Por ejemplo, a través de un análisis de datos de más de 150 años en el área de Randstad en los Países Bajos, se demostró cómo inicialmente las estaciones de tren se ubicaban en zonas ya urbanizadas, pero con el paso del tiempo comenzaron a actuar como núcleos generadores de nuevas áreas construidas(Kasraian et al., 2016). Recientemente, se han planificado estaciones incluso en zonas periféricas con el objetivo de inducir crecimiento urbano en el entorno inmediato de dichas estaciones, configurando un modelo de ciudad conectada y descentralizada. El efecto estructurador del ferrocarril no es exclusivo a los

Países Bajos, en general puede afirmarse que las infraestructuras ferroviarias no solo han respondido al crecimiento urbano, sino que también lo han guiado.

Gracias a la constante evolución tecnológica que ha experimentado, el ferrocarril no ha caído en la obsolescencia, sino que se ha adaptado para satisfacer las demandas del siglo XXI, al igual que satisfacía las demandas de la época cuando originó. Hoy en día, el ferrocarril se sitúa como uno de los modos de transporte principales debido a la introducción de líneas de alta velocidad, trenes eléctricos, digitalización de operaciones, entre otros muchos avances. Se ha vuelto una alternativa real al avión en trayectos de media distancia, al haber logrado reducir significativamente los tiempos de viaje interurbanos, lo cual puede verse ejemplificado con la red AVE en España o el TGV en Francia. Asimismo, en la política de movilidad sostenible de numerosos países, han ganado peso los modelos de planificación urbana que concentran vivienda y servicios en torno a nodos ferroviarios. A estos modelos se les conoce como *transit-oriented development* (TOD), (Alpkokin, 2012).

Más allá de su eficacia operativa, el ferrocarril destaca por sus beneficios medioambientales. A diferencia de otros modos de transporte, el tren, y en particular el tren de alta velocidad presenta un impacto ambiental considerablemente menor. El punto de amortización climática es el tiempo necesario para que las emisiones evitadas por una estructura compensen las emisiones durante su construcción y puesta en marcha. Según el estudio de de Bortoli y Féraille (2024), en el caso del eje París-Burdeos, se estima que, si se reemplazaran los vuelos por alta velocidad ferroviaria, el punto de amortización climática del tren se alcanza en apenas 10 años, frente a los 60 años que serían necesarios si se permite la coexistencia de ambos modos.

En la planificación urbana contemporánea, cada vez son más relevantes aspectos como la mejora de la calidad del aire urbano, la reducción del ruido y la menor ocupación de aspecto vial. Al integrarse fácilmente con redes metropolitanas, el ferrocarril contribuye a descongestionar los accesos urbanos y reducir la necesidad de infraestructuras viales adicionales, trayendo consigo beneficios que no sólo se limitan al plano ambiental, sino que contribuyen al diseño de ciudades más limpias y conectadas. Es por esto que el ferrocarril

aparece como una herramienta clave en consecución de los Objetivos de Desarrollo Sostenible (ODS) 9 (Industria, innovación e infraestructuras) y 11 (Ciudades y comunidades sostenibles).

Para garantizar su seguridad y durabilidad, es de vital importancia el mantenimiento de la red ferroviaria. Ésta se compone de una multitud de infraestructuras, entre ellas, los túneles ferroviarios, los cuales presentan unas condiciones especialmente complejas puesto que son espacios cerrados y de difícil acceso, con iluminación natural reducida y constantemente sometidos a tensiones estructurales. Para prevenir fallos es fundamental la detección y el seguimiento de defectos que pueden aparecer en ellos, como pueden ser las grietas, humedades, desprendimientos o actos vandálicos.

La inspección de túneles ferroviarios suele llevarse a cabo de forma manual, con personal especializado que o recorre los túneles físicamente o evalúa el estado de los mismos visualmente a través de imágenes grabadas, como es el caso de la empresa Axil Ingeniería y Servicios. Este método, aunque históricamente efectivo, presenta múltiples limitaciones. En primer lugar, se trata de una tarea altamente intensiva en tiempo y recursos humanos. Además, estas inspecciones están expuestas a errores subjetivos, fatiga visual y variabilidad entre operarios al depender completamente del ojo humano (Samsami, 2024). En túneles extensos, donde la tarea se traduce en la revisión de miles de imágenes, muchas de las cuales pueden no contener defectos, aumenta la probabilidad de pasar por alto los defectos que sí están presentes.

Si se tiene en cuenta el aumento de número de túneles en explotación debido a políticas de expansión ferroviaria y urbanismo sostenible, lo mencionado previamente cobra aún más relevancia. Ante un mayor número de infraestructuras, aparece una creciente necesidad para automatizar y escalar los procesos de inspección de manera que se puedan mantener niveles adecuados de seguridad sin comprometer los recursos operativos. Es por ello por lo que, al igual que en muchos otros sectores con tediosas tareas realizadas manualmente, el ámbito ferroviario debe respaldarse en la tecnología para incorporar soluciones que permitan digitalizar y automatizar el proceso de inspección.

En este contexto, es importante destacar la colaboración con la empresa Axil Ingeniería y Servicios, dedicada a la captura y explotación de datos mediante tecnologías láser, sensores y visión artificial. Su trabajo abarca desde la inspección de patologías en infraestructuras hasta la generación de modelos 3D, análisis termográficos o inventariados de elementos constructivos. Uno de los puntos señalados directamente por la empresa es la fuerte inversión de tiempo y recursos ligada al análisis de los datos, puesto que dicha tarea continúa siendo manual y visual. En un correo recibido, se menciona explícitamente que "la empresa está buscando algoritmos de detección que permitan automatizar los procesos que más tiempo llevan para el análisis de los datos y optimizar los recursos en el valor añadido." Esta demanda concreta sirve como base para redefinir los objetivos del trabajo:

Objetivo principal

- Validar el uso de la U-Net para la detección automática de defectos en túneles ferroviarios, con el fin de avanzar hacia un análisis automatizado que optimice los recursos y potencie el valor añadido de la inspección

Objetivos específicos

- Entrenar y evaluar el modelo en defectos representativos
- Creación de un *dataset* de imágenes de túneles etiquetadas para realizar investigación sobre la detección de defectos en túneles
- Valorar la aplicabilidad del sistema en los procesos de inspección de la empresa

Capítulo 2. ESTADO DE LA CUESTIÓN

En la literatura, el problema de detección de defectos es un tema que se lleva abordando desde hace años mediante diferentes modelos, conjuntos de datos y métodos de procesamiento de imágenes. A inicios de los 2000, junto a un creciente interés en aplicaciones avanzadas de videovigilancia, se decide buscar la implementación de métodos confiables para reconocer posibles situaciones peligrosas con tasas reducidas de falsas alarmas en errores de detección. En este ámbito, Sacchi et al. (2001) desarrollan un sistema orientado a la detección de actos vandálicos en estaciones de metro no tripuladas. Dicho estudio presenta una arquitectura modular de procesamiento de imágenes compuesta por módulos de bajo nivel, alto nivel y un módulo de compresión de imagen (IU). A bajo nivel se realizan tareas de filtrado de ruido, actualización de fondo y detección de cambios a partir de imágenes de 512x512 píxeles digitalizadas en formato RGB con una tasa de adquisición de 5 fps. Las tareas realizadas a nivel alto permiten describir la evolución temporal de las regiones detectadas en la escena mediante una combinación de técnicas morfológicas y una representación dinámica con grafos.

Inicialmente, el módulo IU se basaba en una base de datos fija de patrones simulados. El papel de dicho módulo es clave puesto que es el que determina si las acciones observadas en la escena son normales o potencialmente peligrosas. Éste se sustituye por una red neuronal perceptrón multicapa (MLP) de tres capas y una capa oculta de 20 neuronas que a partir de diez características geométricas y cinemáticas como pueden ser el perímetro o la velocidad del centro de masa, clasificaba secuencias de vídeos en dos clases: C (calmado) y A (agitado). Esta red fue entrenada con secuencias segmentadas en ventanas de 3 segundos (15 *frames*) simulando escenas de vandalismo. Los resultados obtenidos mostraron una tasa de acierto del 84.2% con falsas alarmas y errores de detección inferiores al 10% aún en escenarios complejos como grupos en movimiento o acciones de grafiti (Sacchi et al., 2001).

En el ámbito de detección de defectos estructurales, en el año 2003 se realiza una comparación sistemática entre cuatro algoritmos de detección de bordes aplicados a imágenes de puentes de hormigón, utilizando un conjunto de 50 imágenes de las cuales la

mitad contenía grietas y la otra mitad no. Los algoritmos utilizados fueron: Sobel, Canny, Transformada Rápida de Fourier y Transformada Rápida de Haar. El estudio concluyó que el último algoritmo mencionado superó significativamente al resto de técnicas en términos de fiabilidad para la identificación de grietas, abriendo paso a métodos más capaces de adaptación al ruido y variaciones de iluminación. Aunque su rendimiento dependía fuertemente del contexto, estos enfoques clásicos fueron pioneros puesto que demostraron que incluso con técnicas deterministas y sin aprendizaje era posible la parcial automatización de tareas de inspección estructural (Abdel-Qader et al., 2003).

En el 2006 surge un enfoque más avanzado y orientado a la automatización total del proceso de inspección por parte de dos de los autores que en el 2003 compararon técnicas clásicas de detección de bordes en hormigón armado. En este trabajo, se propone un sistema de detección no supervisada de grietas en tableros de puentes mediante un algoritmo basado en análisis de componentes principales (PCA). Se evaluaron tres variantes: PCA directo donde se aplica PCA directamente a imágenes completas, PCA tras aplicar realce de líneas y *local PCA* en el cual se aplica el modelo sobre bloques locales de imágenes. En el caso de PCA directo, la imagen global es normalizada y proyectada sobre el espacio definido por los autovectores dominantes generados a partir del conjunto de entrenamiento. Se mide la distancia euclídea entre la proyección de la imagen de prueba y las proyecciones de las imágenes entrenadas y en base al bloque más cercano, se asigna la clase “grieta” o “no grieta”. Este enfoque dio pie a una precisión del 57.5% con 30% de falsas negativas y un 12.5% de falsas positivas (Abdel-Qader et al., 2006).

Por otro lado, la estrategia de *local PCA* inicia con la segmentación de imágenes de 480x480 píxeles en 16 bloques de 120x120 píxeles. Dichos bloques son normalizados y posteriormente proyectados sobre el espacio de autovectores dominantes generados a partir del conjunto de entrenamiento. La clasificación también se realiza mediante distancia euclídea. Con el objetivo de aumentar la sensibilidad a estructuras morfológicas típicas de grietas, se aplican máscaras convolucionales direccionales (vertical, horizontal y oblicuas a ±45°) antes del PCA. El rendimiento del sistema mejora considerablemente con esta estrategia, alcanzando una precisión global del 73%, una reducción de falsas negativas al 12.5% y un aumento de falsas positivas al 15%. En contextos donde se prioriza evitar pasar

por alto una grieta real, este ligero aumento en falsas positivas se considera aceptable (Abdel-Qader et al., 2006).

En el 2012 se propone un sistema híbrido para la detección de grietas en superficies de hormigón que destaca por explorar tanto el análisis de la imagen global como el análisis a nivel de objetos segmentados. Este sistema combina técnicas de procesamiento digital de imágenes con redes neuronales artificiales (ANN) y sistemas de inferencia difusa. El estudio parte de un conjunto de 205 imágenes RGB que fueron procesadas mediante conversión a escala de grises, detección de bordes con operador Sobel y operaciones morfológicas, con el objetivo de limpiar ruido y reconstruir estructuras discontinuas. Posteriormente, se extrajeron dos propiedades claves de cada objeto detectado: el área y la relación entre ejes mayor y menor. Estos valores fueron utilizados como entrada de los modelos clasificadores. En el caso de la red neuronal se diseñó una arquitectura 2-13-1, lo cual describe dos neuronas de entrada, 13 neuronas en la capa oculta y una neurona de salida que produce un valor indicando si el objeto corresponde a una grieta (1) o no (0). La red fue entrenada con retropropagación y función de activación sigmoide. El rendimiento con el modelo ANN en el análisis por objeto fue el mejor (frente a ANN en el análisis global por imagen o el sistema difuso en cualquiera de los dos casos), alcanzando una exactitud del 96.19%. Puesto que la clasificación por objeto evita que regiones limpias de la imagen enmascaren defectos localizados, pudo observarse que este tipo de clasificación resulta más robusta frente al ruido o a la presencia simultánea de múltiples grietas (Choudhary & Dey, 2012)

En el año 2015, se introduce la arquitectura U-Net para tareas de segmentación de imágenes biomédicas. A pesar de desarrollarse en el ámbito biomédico, su relevancia y la versatilidad de la estructura ha permitido que ésta haya sido posteriormente adaptada y aplicada a numerosos campos. Este estudio aborda la problemática de segmentación de estructuras a raíz de un conjunto de tan solo 30 imágenes de microcopia fluorescente. Pese al número limitado de datos, el modelo logró obtener máscaras de segmentación de alta calidad a nivel píxel y una notable capacidad para generalizar, gracias a que las imágenes fueron aumentadas artificialmente mediante técnicas de *data augmentation* (Navab et al., 2015). El estudio demostró que es posible entrenar redes profundas robustas para tareas de segmentación en condiciones reales con escasez de datos. Además, pudo resolver dos

limitaciones claves de los enfoques anteriores como los *Fully Convolutional Networks (FCN)*: la pérdida de resolución espacial y la falta de precisión en bordes.

Con el objetivo de superar las limitaciones de los métodos clásicos de inspección manual o semiautomática, en 2017 se desarrolla un sistema automático de detección de grietas estructurales en estructuras de hormigón basado en redes neuronales convolucionales (CNN). El modelo se entrena con parches de imagen de 256x256 píxeles mediante una metodología *patchwise*, la cual permite su aplicación sobre superficies completas mediante una ventana deslizante, facilitando la detección localizada de defectos en imágenes de gran tamaño. El conjunto de datos se compone de parches distribuidos de manera equitativa entre clases positivas y negativas, donde 40,000 son utilizados para entrenar al modelo y 20,0000 para validación del mismo. La capacidad de generalización del modelo fue mejorada mediante el uso de imágenes contaminadas con ruido realista como son variaciones de iluminación, texturas rugosas o suciedad. La arquitectura CNN utilizada fue de complejidad moderada al constar de tres capas convolucionales, seguidas de capas *max pooling* completamente conectadas, entrenadas con regularización por *dropout*. El estudio demostró la eficacia de entrenar con imágenes realistas, logrando una precisión de 98.2% con una tasa de verdaderos positivos del 99.45% y falsos negativos inferior al 1% (Qu et al., 2018).

Un año después, el problema se aborda con un enfoque híbrido que combina el procesamiento clásico de imágenes con técnicas de aprendizaje profundo. El sistema se divide en dos fases. En primer lugar, se aplica una red neuronal convolucional (CNN) que actúa como filtro inicial para clasificar fragmentos de imágenes como “con grieta” o “sin grieta”. Después, se emplean algoritmos tradicionales de grafos como Dijkstra y árboles de expansión mínima para reconstruir la forma completa de las grietas detectadas. Para evaluar el rendimiento del sistema, el proyecto utiliza dos conjuntos de datos con diferente número y dimensiones de imagen cada uno. A diferencia del resto de publicaciones, las imágenes usadas fueron capturadas por miembros del equipo con una cámara réflex digital Canon 5D, lo cual introduce variabilidad en iluminación, textura y enfoque. Se prueban algoritmos de alta precisión, alta exactitud y una mezcla de ambos, obteniendo distintas precisiones, aunque la mayoría superior al 90%(Qu et al., 2018)

En el mismo año, se vuelve a abordar el problema de detección de grietas en superficies de hormigón utilizando una CNN, aunque este estudio usa como base la arquitectura VGG16, preentrenada en el conjunto ImageNet. El objetivo del estudio es construir un modelo robusto que pueda integrarse en un flujo automatizado de inspección estructural, especialmente a través de imágenes captadas por vehículos áereos no tripulados. El conjunto de datos original constaba de 851 imágenes de 756x756 píxeles que mediante *slicing* dio lugar a un *dataset* final de 3500 imágenes (2336 con grietas y 1164 sin) de bloques de 256x256 píxeles. Las imágenes fueron etiquetadas manualmente. Al utilizar como base la arquitectura VGG16 se superan las limitaciones del reducido tamaño del conjunto de datos mediante *transfer learning.* Del modelo final se fueron modificando tres parámetros: la tasa de aprendizaje, el número de nodos en la última capa densa y el tamaño del conjunto de entrenamiento. El mejor modelo fue resultado de un aprendizaje de 0.10, 32 nodos en la capa final y 100% del conjunto de datos, logrando una precisión del 92.27%. Se observó que mientras que los otros parámetros tuvieron un efecto marginal sobre los resultados, el tamaño del *dataset* tuvo el mayor impacto sobre la precisión (Silva & Lucena, 2018).

En el 2019 de nuevo se aborda este problema de una manera que destaca por su aplicabilidad en entornos no académicos y con recursos limitados. Se utiliza un conjunto reducido de imágenes recolectadas manualmente desde internet y el estudio se centra en construir un *pipeline* funcional con herramientas accesibles como Python, Keras y OpenCV. Mediante técnicas de *data augmentation* como rotación, inversión, *zoom* y desplazamiento, el conjunto inicial se amplió hasta alcanzar un total de 1000 imágenes. El modelo final logró buenos resultados de clasificación binaria y aunque no incluye segmentación por píxel ni evaluación con métricas estandarizadas (precisión, *recall*, IoU), valida el uso de CNN simples entrenadas desde cero (Flor, 2019).

Recientemente, a raíz de una problemática real presentada por el Ayuntamiento de Lisboa, surge un proyecto de investigación que busca la detección en tiempo real de grafitis presentes en toda la ciudad mediante el uso de vehículos equipados con cámaras. El sistema se implementa con imágenes captadas por coches equipados con cámaras y se compone de dos fases. En primer lugar, se entrena un modelo de clasificación basado en ResNet50 con el objetivo de clasificar cada imagen en tres categorías: grafiti ilegal, arte urbano o sin grafiti.

Para representar los tres casos, el conjunto de datos fue cuidadosamente etiquetado y balanceado. Se obtuvo una precisión general del 81.4% y puntuaciones F1 del 86%, 81% y 66% para las clases de arte urbano, grafiti ilegal y sin grafiti. En segundo lugar, se entrenó un modelo de detección de objetos utilizando Faster R-CNN con ResNet50 y FPN como *backbone* con el propósito de detectar las coordenadas del grafiti en la imagen mediante una *bounding box* y asignarle su clase correspondiente. Se alcanzó una intersección sobre unión (IoU) del 70,3%. Además, durante el entrenamiento de los modelos se aplicaron técnicas clave como *data augmentation* y *transfer learning* (Fogaça et al., 2023).

Los modelos analizados ofrecen una base técnica sólida y versátil sobre la que construir sistemas de detección adaptados a defectos visuales presentes en túneles ferroviarios. A pesar de que la mayoría de los estudios mencionados se centran en la detección de grietas en estructuras de hormigón, las técnicas empleadas son agnósticas al tipo de defecto y pueden adaptarse a otros contextos con una representación visual clara del fenómeno a detectar, como pueden ser los grafitis, las humedades, eflorescencias o fisuras. La Tabla 1 resume las diferentes soluciones analizadas en este capítulo.

Título	*A neural network approach to vandalism detection in metro stations*	*Edge detection techniques for crack identification in concrete structures*	*Principal component analysis for crack detection in concrete bridge decks*	*Hybrid image processing and ANN approach for crack detection in concrete*	*Convolutional networks for biomedical image segmentation*
Año	2001	2003	2006	2012	2015
Defecto a identificar	Actos vandálicos, grafitis	Grietas en hormigón	Grietas en tableros de puente	Grietas en hormigón	Segmentación biomédica
Tipo de red	Procesamiento modular + MLP	Algoritmos clásicos (Sobel, Canny, FFT, Haar)	PCA directo/local + máscaras bidireccionales	Procesamiento clásico + ANN + lógica difusa	U-Net original
Número de imágenes	-	50	-	205	30
Data augmentation	No	No	No	No	Sí
Resolución	512x512	-	480x480 (bloques 120x120)	-	512x512
Resultados	84.2%	Haar > resto	Directo: 57.5% Local PCA: 73%	ANN por objetos: 96.19%	-

Concrete Surface crack detection using deep learning	*A crack detection approach based on CNN and graph algorithms*	*Crack detection in UAV images using transfer learning with VGG16*	*Surface crack detection with simple CNN pipeline*	*Automatic graffiti detection in Lisbon using deep learning*
2017	2018	2018	2019	2023
Grietas en hormigón	Grietas en hormigón	Grietas en hormigón (UAV)	Grietas	Grafitis
CNN *patchwise*	CNN inicial + reconstrucción con grafos	CNN con *transfer learning*	CNN simple	ResNet50 + Faster R-CNN
40000	D1: 100 D2: 118	851	67	-
No	No	Sí (*slicing*)	Sí	Sí
256x256	D1: 400x300 D2: 480x320	756x756 (recortadas a 256x256)	-	-
98.2%	≥90%	92.27%	-	Clasificación: 81.4% Detección: IoU= 70.3%

Tabla 1. Comparación de los estudios revisados

Capítulo 3. FUNDAMENTOS TEÓRICOS

3.1 INTRODUCCIÓN A LA SEGMENTACIÓN DE IMÁGENES

El aprendizaje automático, comúnmente conocido como *machine learning*, es una rama principal de la inteligencia artificial orientada al desarrollo de sistemas capaces de aprender patrones a partir de datos sin requerir instrucciones explícitas. Mientras que en la programación tradicional el comportamiento del sistema es definido por reglas concretas diseñadas por el programador, en el aprendizaje automático se parte de un conjunto automático y es el propio sistema el que infiere las reglas internas que le permiten resolver una tarea. En el ámbito de la visión informática (*computer vision)* resulta especialmente relevante ya que, debido a la complejidad y variabilidad de las imágenes en el mundo real, las reglas manuales son difíciles de definir.

Existen varios tipos de aprendizaje automático, según el tipo de datos disponible y el objetivo de la tarea. El más común y el más relevante para este proyecto es el aprendizaje supervisado, el cual se basa en el uso de ejemplos etiquetados. Cada entrada del conjunto de datos tiene asociada una salida esperada, llamada etiqueta o *ground truth*. A partir de la entrada, el modelo aprende a predecir la salida correcta, ajustando sus parámetros internos según corresponda para minimizar los errores cometidos durante el entrenamiento. Contrariamente, en el aprendizaje no supervisado no se proporcionan etiquetas, sino que el objetivo es descubrir patrones ocultos o estructuras dentro de los datos, como agrupaciones o correlaciones. El aprendizaje por refuerzo tampoco utiliza etiquetas, el proceso de aprendizaje se basa en la interacción continua entre un agente y un entorno. El agente toma decisiones y recibe retroalimentación en forma de recompensas o penalizaciones, debe explorar y aprender por sí mismo qué acciones le conducen a mejores resultados a largo plazo.

Como se ha mencionado previamente, el aprendizaje supervisado se caracteriza por el uso de datos etiquetados. Cada ejemplo del conjunto de datos incluye tanto la información

que se desea analizar como la salida correcta que el modelo desea analizar. La forma de relacionar las entradas con las salidas correspondientes es mediante el uso de una función de pérdida. Una función de pérdida cuantifica el error cometido en cada predicción, durante el entrenamiento del modelo se busca minimizar esta función a través de un proceso iterativo de ajuste de parámetros. El ajuste de parámetros normalmente se lleva a cabo utilizando algoritmos de optimización (e.g. descenso del gradiente, *gradient descent)* que se encargan de progresivamente modificar los pesos del modelo de tal forma que se minimice el valor de la función de pérdida (Chen, 2019).

En el aprendizaje supervisado, es habitual dividir el conjunto de datos en tres subconjuntos: uno de entrenamiento, otro de validación y otro de prueba. El conjunto de entrenamiento es utilizado como bien indica su nombre, para el entrenamiento del modelo, fase en la cual el modelo optimiza su función de pérdida ajustando sus parámetros según aprende a partir de los ejemplos etiquetados. El conjunto de validación se usa para evaluar el rendimiento del modelo, ajustar hiperparámetros y detectar posibles problemas como el sobreajuste (*overfitting*). Aunque el conjunto de validación es una buena práctica, no es un requisito formal del aprendizaje supervisado y existen varias razones por las cuales no siempre es incluido, por ejemplo, si se dispone de pocos datos etiquetados. Completamente apartado del proceso de entrenamiento y ajuste, se encuentra el conjunto de prueba, el cual permite evaluar con objetividad la capacidad del modelo para hacer predicciones sobre datos completamente nuevos. El objetivo de dividir los datos de tal forma es evitar que el modelo los memorice y poder verificar que éste es capaz de generalizar su conocimiento a datos no observados previamente.

La segmentación de imágenes es una tarea que tiene como finalidad asignar una etiqueta a cada uno de los píxeles que componen una imagen, clasificándolos en función de su pertenencia a diferentes objetos, estructuras o áreas de interés. Mientras que en la clasificación de imágenes el modelo asigna una única etiqueta a la imagen completa y responde a la pregunta "¿qué hay en esta imagen?", la segmentación busca un análisis espacial detallado que permita identificar qué elementos hay y la localización espacial de dichos elementos. La segmentación de imágenes puede entenderse como una extensión

natural del aprendizaje supervisado clásico, en el que el objetivo es generar una salida estructurada, lo cual implica que cada píxel actúa como una unidad de decisión dentro del aprendizaje y el modelo debe aprender a etiquetarlos correctamente en función de su contexto visual. Luego la salida del modelo es una interpretación semántica detallada de la escena, una máscara densa en la que se especifica para cada píxel a qué clase pertenece, no una etiqueta única para la imagen total.

En función del tipo de etiquetado que se aplica a los píxeles, se tiene un tipo u otro de segmentación de imágenes. Si cada píxel se clasifica como perteneciente a una clase de interés o al fondo, se trata de segmentación binaria, la cual es útil en aplicaciones donde se busca detectar un único tipo de objeto. La salida del modelo es una máscara de un solo canal con valores binarios. Sin embargo, si cada píxel recibe una única etiqueta entre un conjunto de posibles clases mutuamente excluyentes, se trata de segmentación multiclase. Si en cambio, un mismo píxel pudiese pertenecer a más de una clase al mismo tiempo, se trataría de segmentación *multilabel*. Este tipo de segmentación es aplicable cuando las clases representan atributos no excluyentes o hay solapamientos y la salida del modelo es un tensor de múltiples canales, donde cada canal representa la probabilidad de presencia de esa clase en cada píxel.(Keylabs, 2024)

Existen múltiples enfoques para abordar la segmentación de imágenes, partiendo de métodos clásicos basados en umbralización, detección de bordes o técnicas de *clustering* hasta aproximaciones más recientes fundamentadas en arquitecturas de *deep learning*. Este apartado se centra en la familia de modelos U-Net, ya que están diseñados para generar segmentaciones densas y han demostrado un rendimiento especialmente sólido en escenarios donde se requiere una localización espacial precisa de los objetos de interés, como es el caso de la detección de defectos en túneles ferroviarios.

3.2 ARQUITECTURA U-NET

Las tareas de segmentación de imágenes requieren modelos capaces de procesar imágenes tanto en términos de contenido, como en cuanto a su estructura espacial. Debido a

su capacidad para extraer y jerarquizar características visuales de forma eficiente, durante lo últimos años se han consolidado las redes neuronales convolucionales (CNNs) para abordarlas. Muchas arquitecturas clásicas de CNN están diseñadas principalmente para tareas de clasificación, puesto que la segmentación requiere una salida densa y estructurada, este requisito ha motivado el desarrollo de arquitecturas que amplían la estructura básica de las CNNs para adaptar su funcionamiento a este objetivo.

En el año 2015, en el congreso *Medical Image Computing and Computer-Assisted Intervention* (MICCAI) se presenta una arquitectura de red neuronal convolucional diseñada específicamente para tareas de segmentación semántica de imágenes llamada U-Net (Ronneberger et al., 2015) . En ese momento a pesar de que las CNNs, con modelos como AlexNet o VGG, ya habían demostrado un rendimiento excelente en tareas de clasificación, no estaban optimizadas para producir salidas estructuradas a nivel de píxel. El desafío al que se enfrentaban los autores de esta nueva arquitectura la segmentación precisa de estructuras celulares en imágenes de microscopía, con un conjunto de datos reducido y altamente especializado, se propone esta nueva arquitectura como solución.

Las arquitecturas existentes no proporcionaban una solución adecuada a dicho desafío por dos motivos principales. En primer lugar, a través de capas de *pooling* (operaciones que disminuyen el tamaño de las representaciones conservando solo la información más relevante como el valor máximo o medio en cada región), se eliminaba progresivamente la resolución espacial impidiendo recuperar detalles finos. Por otro lado, para entrenar con eficacia requerían grandes volúmenes de datos lo cual no era viable en entornos clínicos. Para superar estas limitaciones, los autores diseñaron una arquitectura simétrica de tipo *encoder-decoder* con conexiones de salto entre sus etapas que permitiera combinar el contexto semántico aprendido en capas profundas con la precisión espacial de las capas superficiales.

Según Ronneberger, Fischer y Brox (2015) la U-Net se compone de las tres partes fundamentales mencionadas en el párrafo anterior. El *encoder* o bloque de codificación se encarga de capturar el contexto semántico de la imagen. Está compuesto de una serie de

bloques convolucionales seguidos de operaciones de *downsampling* que reduce la resolución espacial de la imagen. El muestreo descendiente (*downsampling*) consiste en aplicar operaciones que eliminan información redundante a nivel espacial, conservando únicamente los valores más representativos de cada región. Conforme se avanza por la red, disminuye el tamaño espacial de las representaciones, por ejemplo, de 256x256 a 128x128 píxeles, pero aumenta la profundidad, es decir, el número de canales. Esto permite al modelo aprender características cada vez más abstractas y complejas.

La función del bloque de decodificación (*decoder*) es reconstruir la resolución original a partir de las representaciones comprimidas generadas por el *encoder*. Emplea operaciones de *upsampling* que son contrarias a las de *downsampling* y permiten recuperar el detalle perdido según aumentan progresivamente el tamaño espacial de las activaciones. El *upsampling* o muestreo ascendente, busca expandir las características aprendidas para que coincidan con la escala original de la imagen y se pueda generar una máscara de segmentación detallada.

Otro componente esencial de la arquitectura U-Net son las conexiones de salto o *skip connections*, las cuales permiten copiar directamente las activaciones generadas en las capas del *encoder* y concatenarlas en niveles simétricos con las del *decoder*. Como consecuencia del *downsampling*, se suele perder la información espacial fina (bordes, contornos o texturas), por lo que el objetivo de estas conexiones es preservarla. La capacidad del modelo para segmentar con precisión, aún en objetos pequeños o con bordes poco definidos, mejora significativamente al combinar características profundas con características superficiales.

3.2.1 FLUJO DE DATOS

El funcionamiento interno de la arquitectura U-Net puede comprenderse observando el flujo de datos a lo largo de la red, desde la imagen de entrada hasta la generación de la máscara segmentada. Este recorrido se articula en torno a dos caminos simétricos, uno de contracción y otro de expansión, conectados por un cuello central y reforzados mediante conexiones de salto. La Figura 1 ilustra este proceso. A continuación, se describe cómo se transforma la información a lo largo de la red.

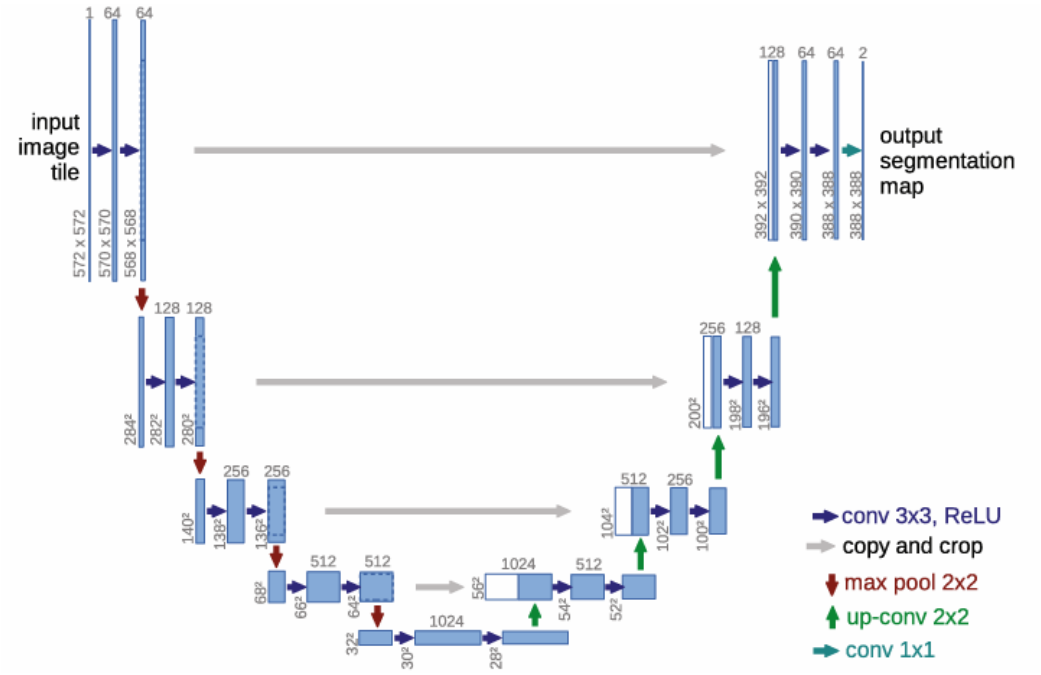


Figura 1. Arquitectura original de la red U-Net. Fuente: Ronneberger, O., Fischer, P., & Brox, T. (2015)

El proceso comienza en la parte superior izquierda, donde se introduce una imagen de entrada de 572x572 píxeles y 1 canal. Esta imagen pasa por dos convoluciones 3x3 con activación ReLU, como indican las flechas azules, produciendo 64 mapas de características. Representado por la flecha roja, se aplica una operación de *max pooling* 2x2 que reduce la resolución a la mitad. Mientras que la resolución disminuye de 572x572 a 284x284, el número de canales se duplica y de 64 aumenta a 128. Cada nivel del *encoder* consta de dos convoluciones y una operación de *max pooling*. Este bloque se repite en varias etapas, en cada una, la resolución se reduce mientras que el número de canales se duplica, permitiendo al modelo aprender representaciones más abstractas.

En el centro de la red, denominado cuello de la red (*bottleneck*), se alcanza el punto de mínima resolución espacial y máxima profundidad semántica, que para este ejemplo sería 28x28 píxeles y 1024 canales. Además, se aplican dos nuevas convoluciones 3x3 que permiten al modelo integrar la información aprendida a nivel global antes de comenzar la reconstrucción. A continuación, está el *decoder*, a partir del cual el modelo entra en la fase

reconstrucción. A cada bloque de esta fase le corresponde una operación de *up-conv* 2x2 que duplica la resolución espacial indicada por las flechas verdes, indicada por las flechas grises una concatenación con la salida correspondiente del *encoder* que transfiere los mapas de características de bajo nivel mediante copiado y recorte y finalmente, dos convoluciones 3x3 para refinar la información combinada(Analytics Vidhya, 2025) .

Una vez finalizado el proceso, se obtiene un mapa de características de tamaño 388x388x64 al cual, para transformarlo en una máscara de segmentación, se le aplica una convolución 1x1 indicada por la flecha azul claro. Esta convolución reduce la profundidad del tensor al número de clases objetivo que en el ejemplo mostrado en la figura es 2. En el caso de segmentación binaria, la salida sería un solo canal con activación *sigmoid,* mientras que en segmentación multiclase habría un canal por clase y activación *softmax*. Debido a la pérdida de bordes en cada convolución sin *padding,* la resolución de la salida es ligeramente menor que la de la entrada original. Este efecto suele corregirse en implementaciones modernas usando *same padding* o recortes apropiados para que la salida y la entrada coincidan (Kerem Turgutlu, 2018).

3.3 ENTRENAMIENTO DE UNA U-NET

Una vez definida la arquitectura de la red U-Net, el siguiente paso es su entrenamiento. Al ser un modelo de aprendizaje supervisado, su entrenamiento se basa en un conjunto de datos etiquetados, compuesto por pares (X, Y) donde X son las imágenes de entrada, normalmente de un tamaño fijo, y las máscaras correspondientes son representadas por la Y. Para un entrenamiento efectivo la calidad y precisión de las máscaras es fundamental ya que el modelo aprende directamente a partir de ellas. Como también se ha explicado previamente, es común dividir el conjunto de datos en tres subconjuntos (entrenamiento, validación y prueba).

Un elemento esencial del entrenamiento es la función de pérdida, ya que la minimización de dicha función es el mecanismo mediante el cual el modelo aprende. La función cuantifica el error entre las predicciones generadas por la red y las etiquetas (*ground*

truth) del conjunto de entrenamiento. Mediante el algoritmo de *backpropagation*, este error se propaga hacia atrás a través de la red y los pesos se actualizan gradualmente gracias a un optimizador (como Adam o SGD) para en las siguientes iteraciones reducir dicho error. Por lo tanto, elegir adecuadamente una función de pérdida es una decisión. Existen varias funciones y varían según el tipo de segmentación que se trate, siendo la segmentación binaria la relevante en este proyecto.

En segmentación binaria, las funciones más comunes son: *Binary Cross Entropy (BCE), Dice Loss* y *Tversky Loss*. BCE calcula el error entre la predicción del modelo, que es una probabilidad entre 0 y 1, y el valor real (0 o 1) para cada píxel de forma independiente. Tiene varias ventajas como el hecho de que es una función de pérdida bien establecida y ampliamente compatible con optimizadores o que es interpretable como *log-loss* probabilístico, sin embargo, padece de sensibilidad al desbalanceo de clases y tiende a favorecer el fondo cuando la clase positiva es poco frecuente (Terven et al., 2025). Más eficaz ante el desbalanceo de clases está la función *Dice Loss.* Basada en el coeficiente *Dice,* esta función mide la superposición entre la máscara predicha y la real, penalizando más los errores en objetos pequeños. Sus limitaciones incluyen menos estabilidad durante las primeras épocas de entrenamiento y que no penaliza igual todos los tipos de error (Terven et al., 2025). Aún más útil en segmentación con clases muy desbalanceadas es la *Tversky Loss,* la cual generaliza la *Dice Loss* mediante la introducción de dos parámetros que permiten controlar el peso relativo de los falsos positivos y falsos negativos (Robin Vinod, 2020). En la práctica, es común que las implementaciones combinen BCE para aprovechar su estabilidad con *Dice* o *Tversky* y así beneficiarse también de la sensibilidad a la superposición de estos otros dos tipos de funciones de pérdida.

La función de pérdida es minimizada mediante un algoritmo de optimización. Adam es el más habitual, siendo una versión adaptativa del descenso del gradiente que ajusta automáticamente la tasa de aprendizaje para cada parámetro. La tasa de aprendizaje es un valor importante a definir ya que influye en la velocidad de convergencia, valores típicos pueden ser 1e-4 o 1e-5 (Jadon, 2020). Otros valores que son importantes definir son el número de épocas (*epochs*), es decir, la cantidad de ciclos completos sobre el conjunto de

entrenamiento, y el *batch size* que es el número de imágenes procesadas simultáneamente. Este último valor depende de la memoria disponible en la GPU.

El rendimiento del modelo se evalúa durante y después del entrenamiento, usando métricas específicas. Una de las más comunes *accuracy* que indica la proporción de píxeles correctamente clasificados. *Intersection over Union* (IoU) mide la superposición entre predicción y *ground truth,* aunque el *Dice coefficient* es similar, se diferencia de IoU ya que pondera más los aciertos en clases pequeñas. Para detectar posibles signos de sobreajuste, también es común evaluar la pérdida en validación. También pensado para evitar problemas como el sobreentrenamiento, es usual emplear funciones de control conocidas como *callbacks (Müller et al., 2022)*. Por ejemplo, *early stopping* detiene el entrenamiento si tras varias épocas la métrica de validación no mejora, *model checkpoint* guarda el modelo con mejor rendimiento en validación y *ReduceLROnPlateau* disminuye la tasa de aprendizaje si la pérdida se estabiliza.

En resumen, entrenar una red U-Net no consiste únicamente en disponer de datos etiquetados correctamente sino de tomar decisiones informada sobre otros elementos clave como son la función de pérdida, el optimizador y las técnicas de regularización.

3.4 TIPOS DE U-NET

Desde su origen en 2015, la arquitectura U-Net ha sido modificada de numerosas formas con el fin de mejorar su rendimiento, adaptarla a otros dominios o solucionar limitaciones específicas. La pérdida de información espacial, la sobrecarga computacional o la sensibilidad al desbalanceo de clases son tan sólo algunos ejemplos de estas limitaciones. Independiente de los cambios introducidos por las nuevas variantes, todas mantienen la estructura general de codificación, decodificación y conexiones de salto. La U-Net estándar es ideal para segmentación binaria o multiclase, siendo eficiente para imágenes de tamaño variable gracias a que carece de capas densas (*fully connected*). Aunque es muy utilizada en biomedicina, ámbito en el que se originó, también es empleada en inspección industrial, en el mundo de la agricultura o la detección.

La U-Net clásica, especialmente en imágenes con estructuras complejas o con fondos muy dominantes, es culpable de introducir ruido o información irrelevante ya que todas las activaciones del *encoder* se copian directamente al *decoder* sin distinción. Con el fin de filtrar la información que se transfiere entre estos dos componentes, surge la Attention U-Net. Esta variación añade módulos de atención (*attention gates*) en las conexiones saltos que aprenden a asignar pesos dinámicos a las regiones del mapa de características antes de concatenarlas con el *decoder*. En casos con objetos pequeños o difíciles de detectar, estas modificaciones resultan extremadamente útiles ya que estas compuertas permiten que el modelo ignore regiones no informativas y realce las áreas de interés. Al no propagar información irrelevante, esta arquitectura reduce los falsos positivos (Oktay et al., 2018). Debido a ello, suele aplicarse en la segmentación de estructuras de bajo contraste o bajo tamaño relativo y para la segmentación médica avanzada como es el caso de los tumores o las lesiones cerebrales. Su arquitectura se muestra a continuación:

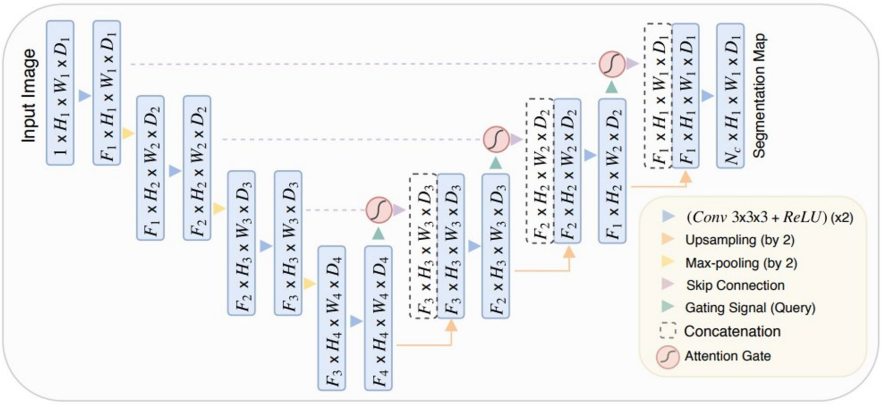


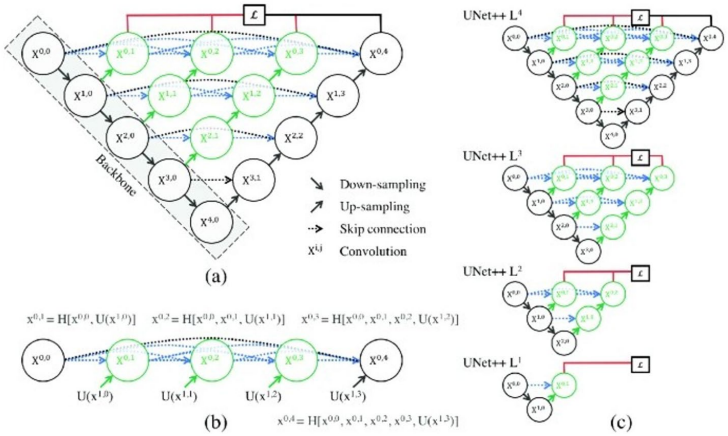
Figure 1: A block diagram of the proposed Attention U-Net segmentation model. Input image is progressively filtered and downsampled by factor of 2 at each scale in the encoding part of the network (e.g. $H_4 = H_1/8$). N_c denotes the number of classes. Attention gates (AGs) filter the features propagated through the skip connections. Schematic of the AGs is shown in Figure 2. Feature selectivity in AGs is achieved by use of contextual information (gating) extracted in coarser scales.

Figura 2. Esquema de Attention U-Net. Fuente: Oktay et al., 2018.

Mientras que en la U-Net clásica las conexiones de salto conectan directamente niveles simétricos del *encoder* y *decoder*, hay una variante llamada U-Net++ en la cual se intercalan múltiples convoluciones intermedias en rutas densas (*nested skip connections*), generando subredes internas que permiten una fusión progresiva y refinada de características

(véase Figura 3). Esta arquitectura reduce el desfase de información entre las capas profundas y superficiales al mejorar la integración semántica entre niveles de diferente resolución, sin embargo, el incremento de parámetros y profundidad trae consigo un mayor coste computacional. En modelos donde se busca una mayor robustez semántica sin sacrificar precisión, es habitual usar esta variante.

Fig. 1.



(a) UNet++ consists of an encoder and decoder that are connected through a series of nested dense convolutional blocks. The main idea behind UNet++ is to bridge the semantic gap between the feature maps of the encoder and decoder prior to fusion. For example, the semantic gap between ($X^{0,0}$, $X^{1,3}$) is bridged using a dense convolution block with three convolution layers. In the graphical abstract, black indicates the original U-Net, green and blue show dense convolution blocks on the skip pathways, and red indicates deep supervision. Red, green, and blue components distinguish UNet++ from U-Net. (b) Detailed analysis of the first skip pathway of UNet++. (c) UNet++ can be pruned at inference time, if trained with deep supervision. (Color figure online)

Figura 2. Arquitectura de la U-Net++. Fuente: Stoyanov et al., 2018.

En contextos donde los datos son ruidosos, el entrenamiento es difícil o se requiere mayor profundidad sin pérdida de rendimiento (e.g. defectos industriales con bordes irregulares o análisis remoto), es común utilizar otra variante conocida como ResUNet, mostrada en la Figura 4. Combinando la arquitectura U-Net con los principios de las redes residuales (ResNet), esta técnica facilita el flujo de gradientes durante el entrenamiento, especialmente en redes profundas, y mitiga el problema de la desaparición del gradiente. Según Zhou et al. (2020) esta variante incorpora bloques residuales, dentro de los módulos convolucionales del *encoder* y *decoder*, que permiten al modelo aprender funciones residuales (la diferencia entre la entrada y la salida esperada de una capa) en lugar de

aprender directamente la función completa. La ResUNet, al mantener conexiones internas dentro de cada bloque, permite una mayor estabilidad durante el aprendizaje y mejora la capacidad del modelo para aprender representaciones más ricas y robustas.

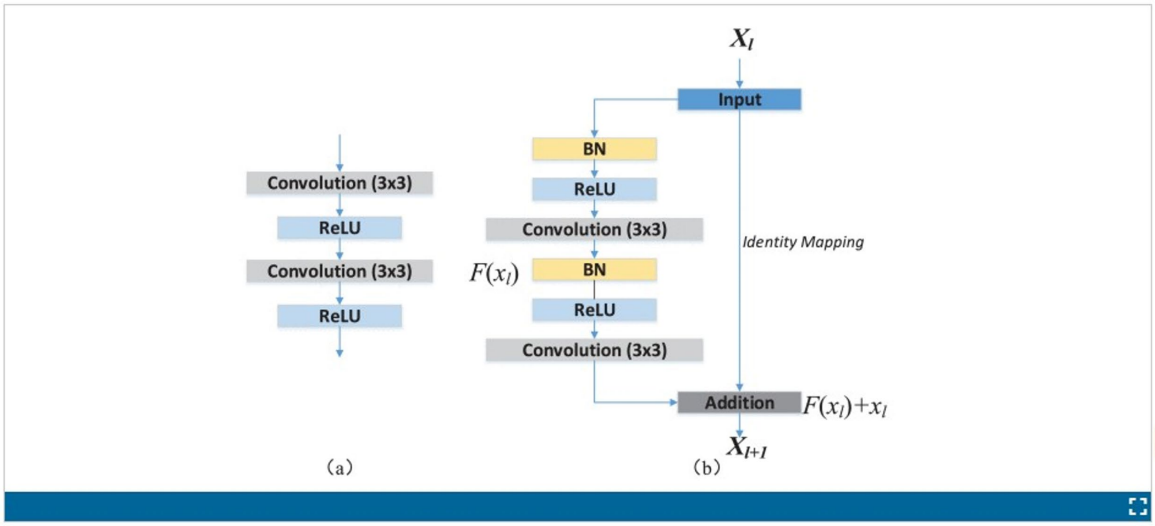


Fig. 1.
Building blocks of neural networks. (a) Plain neural unit used in U-Net. (b) Residual unit with identity mapping used in the proposed ResUnet.

Figura 3. Arquitectura de la ResUNet. Fuente: Zhang et al., 2018.

El concepto original de la U-Net puede extenderse al dominio tridimensional mediante la adaptación al espacio 3D de todas las operaciones convolucionales de *pooling* y *upsampling*, dando lugar a la arquitectura 3D U-Net. Además de usar operaciones de *pooling* y *upsampling* tridimensionales, la estructura mantiene la simetría *encoder-decoder* y utiliza convoluciones 3D. Este modelo procesa volúmenes de datos, en lugar de trabajar con imágenes bidimensionales (Çiçek et al., 2016). Su capacidad para conservar las relaciones espaciales en las tres dimensiones (alto, ancho y profundidad) es especialmente relevante en aplicaciones donde la información volumétrica es clave, como la segmentación de órganos en resonancias magnéticas o tomografías computarizadas, o el análisis de datos geoespaciales o LIDAR (*Light Detection and Ranging*). El uso de esta variante suele requerir hardware especializado y técnicas de reducción de complejidad como recortes (*patching*) o redes híbridas 2.5-f debido a su alto coste computacional y de memoria.

Finalmente, con el objetivo de mejorar la U-Net clásica en términos de robustez multiescala y eficiciencia de aprendizaje, se propone la MultiResUNet, planteada por Ibtehaz

& Rahman (2019) . Esta arquitectura introduce dos componentes clave: los bloques MultiRes y los bloques ResPath. Los primeros consisten en convoluciones agrupadas con diferentes tamaños de *kernel* (e.g. 3x3, 5x5, 7x7), lo cual permite capturar patrones visuales a distintas escalas simultáneamente. Por otro lado, los bloques ResPath facilitan la transmisión de información relevante a lo largo de la red y mejoran la propagación del gradiente puesto que actúan como rutas de aprendizaje residual entre el *encoder* y *decoder*. Esta variante de la U-Net resulta eficaz en contextos como la segmentación de imágenes naturales complejas, imágenes de satélite o análisis detallado de estructuras biológicas con una significativa variabilidad morfológica.

Desde su formulación original, la arquitectura U-Net ha continuado adaptándose a un amplio abanico de problemas y dominios. Cada una de sus variantes responde a desafíos específicos y su elección debe basarse en las características del problema a resolver, los recursos disponibles y la naturaleza de los datos. Este trabajo opta por la implementación de la U-Net clásica, debido a su eficacia demostrada como por su menor complejidad computacional. Esta decisión no considera únicamente los criterios metodológicos sino también la limitación de recursos disponibles, particularmente la memoria RAM del equipo local y la capacidad disponible, así como el tiempo restringido y el alcance definido de este Trabajo de Fin de Grado. Es una implementación inicial que busca establecer una base sólida, teniendo como objetivo establecer una prueba de concepto funcional y realista de manera que, en un futuro, si los resultados son prometedores y de interés para la empresa, el sistema podría escalarse haciendo uso de una infraestructura más potente y aplicándose a mayor escala, abarcando todas las necesidades operativas de Axil Ingeniería.

Capítulo 4. DESCRIPCIÓN DE TECNOLOGÍAS

El proyecto se implementó en Python, un lenguaje de programación que en los últimos años se ha consolidado como opción predeterminada en el ámbito de la ciencia de datos y el aprendizaje automático (Craig Hale, 2025). Su sintaxis sencilla que facilita la rápida construcción de prototipos, así como una amplia comunidad que mantiene un ecosistema rico de librerías científicas y de *deep learning* son sus principales ventajas.

4.1 LIBRERÍAS PRINCIPALES

Como se ha mencionado, una de las principales ventajas de Python son sus librerías, dentro de las cuales cada una pudo usarse para funciones específicas en el flujo de trabajo.

- OpenCV: permitió realizar operaciones básicas de lectura, escritura y modificación de ficheros. También sirvió para la aplicación de transformaciones morfológicas como erosión o dilatación. Esta librería se empleó como herramienta principal de procesamiento de imágenes, ya que sus operaciones fueron necesarias para generar y refinar las máscaras de los defectos.
- NumPy: constituyó la base de la manipulación de los tensores previos al entrenamiento. En forma de *arrays* multidimensionales, proporcionó estructuras de datos eficientes sobre los cuales fue posible realizar cálculos necesarios en la preparación del conjunto de datos.
- Matplotlib: su capacidad para generar gráficos y figuras permitió evaluar cualitativamente el rendimiento y comprobar de manera visual los efectos de las distintas etapas de preprocesamiento. Se utilizó para la visualización de resultados y para la comparación entre las predicciones del modelo y las máscaras reales.
- *Python Imaging Library* (PIL): se empleó como complemento para la gestión y manipulación básica de imágenes. Facilitó la apertura, conversión y transformación de ficheros en distintos formatos.

- TensorFlow: la plataforma de cálculo principal para implementar y entrenar las redes. Permitió ejecutar de forma eficiente los cálculos intensivos asociados al entrenamiento de modelos gracias a su soporte para operaciones con tensores y a su compatibilidad con GPU.
- Keras: facilitó la implementación de la U-Net, así como la gestión de parámetros de entrenamiento. Se utilizó como *Application Programming Interface* (API) integrada en TensorFlow para definir y configurar las arquitecturas de manera más sencilla y modular.

4.2 RECURSOS HARDWARE

En una fase inicial, el proyecto se desarrolló en cuadernos Jupyter ejecutados desde Anaconda Navigator en un ordenador local, resultando útil para la preparación de los datos y la implementación de los primeros *scripts*. De cara al desarrollo posterior del proyecto, este entorno presentaba una limitación fundamental: la imposibilidad de aprovechar unidades procesamiento gráfico (GPUs), sin las cuales el entrenamiento de las redes neuronales resultaría excesivamente lento. Por esta razón, se optó por migrar los datos a Google Drive que podía integrarse directamente con Google Colab, una plataforma en la nube que permite ejecutar cuadernos Jupyter con acceso a GPUs. Debido a que el plan gratuito ofreció recursos de GPU limitados, tiempos de sesión restringidos y desconexiones aleatorias, fue necesario recurrir a la contratación de servicios de pago. El coste total asumido ascendió a 48, 24€, desglosado de la siguiente forma:

- 3,48€ en Google Drive, correspondientes a una ampliación inicial de almacenamiento a 100 GB y una posterior ampliación a 200 GB
- 44,76€ en Google Colab, destinados a la adquisición de 400 unidades de cómputo para acceder a GPUs dedicadas

Capítulo 5. TRABAJO DESARROLLADO

Axil Ingeniería y Servicios, la empresa colaboradora, proporcionó un amplio volumen de datos, obtenidos mediante perfilómetros láser. Éstos son dispositivos en los que un haz fijo se proyecta contra un espejo en rotación para capturar una sección de 360º. Conforme el equipo avanza longitudinalmente por el túnel, va generando secciones sucesivas que, al combinarse, permiten reconstruir la geometría completa de la estructura con gran nivel de detalle.

5.1 PROCESAMIENTO DE DATOS

El conjunto de datos recibido inicialmente estaba compuesto de tres carpetas: PAT, TIR y VIS. Cada una de estas carpetas contenía cientos de PDFs, cada uno con distinto número de páginas. La carpeta VIS contenía las imágenes originales de los túneles, mientras que la carpeta PAT representaba las patologías mediante polígonos de colores superpuestos. La carpeta TIR contenía las imágenes infrarrojas las cuales no eran necesarias para el desarrollo de este proyecto y es por ello por lo que esta última carpeta no fue utilizada en absoluto.

La resolución de las imágenes en los PDFs era extremadamente alta por lo que conservarla fue primordial. Inicialmente se extrajeron las imágenes en su totalidad de los PDFs, optando por un formato sin pérdidas como PNG para no degradar la calidad de los datos. La conversión se realizó a un DPI (puntos por pulgada) fijo que representa la resolución de la imagen. Mantener un valor constante de DPI en todas las páginas fue fundamental para evitar desalineaciones provocadas por diferencias de resolución entre documentos. De esta manera, cada imagen quedó perfectamente estandarizada e identificada mediante el tipo de documento (ORIGINAL o PAT), el fichero al que correspondía (e.g. 11113) y el número de página dentro de ese fichero (page_id). El *pipeline* de procesamiento posterior fue simplificado gracias a esta homogeneización inicial.

5.1.1 CREACIÓN DE LAS MÁSCARAS

A partir de las imágenes extraídas de los ficheros dentro de PAT, donde cada defecto está codificado por un color y un patrón de referencia, se generaron las máscaras binarias por patología. La leyenda de patologías proporcionada por la empresa se muestra a continuación:

Figura 4. Leyenda de patologías.

Como puede observarse en la Figura 5, la leyenda de patologías incluía una gran variedad de defectos estructurales y superficiales, cada uno representado en los ficheros PAT por un color concreto o un tono dentro de una misma gama cromática. Aparte de manejar un amplio volumen de datos, un reto añadido era el hecho de que varios defectos se distinguían únicamente por matices sutiles de color o por patrones de relleno diferentes (sólidos, rayados, etc.). Intentar abarcar todas las patologías simultáneamente era un reto prácticamente imposible teniendo en cuenta que los rangos de color en el espacio digital no son valores únicos sino intervalos que fácilmente se cruzan entre sí, generando una fuerte tendencia al solapamiento cromático. Debido a esto, se tomó la decisión clave de centrar el proyecto en únicamente dos patologías representativas:

1. Rosa: grafitis
2. Azul: todos los defectos asociados a humedades, independientemente de su tipología concreta.

Inicialmente se consideró un tercer grupo que incluía color rojo, pero tras realizar varias pruebas de generación de máscaras, no había forma de evitar que detectara adecuadamente el rojo sin detectar también el rosa, por lo que se optó por tener únicamente en consideración los dos grupos que aparecen listados. Además, en vez de utilizar una única función para la generación de todas las patologías, se decidió definir funciones específicas de generación de máscaras para cada color. De esta manera el método de generación de máscaras podía ser ajustado según las particularidades de cada defecto y se redujo la tasa de errores sistemáticos que habría provocado un enfoque único y global.

Aunque el detalle de la implementación informática no se considera el núcleo de este trabajo, se ha considerado oportuno incluir en los Anexos el código de las funciones de generación de máscaras. En todos los casos se aplicó inicialmente un *kernel* morfológico pequeño con el propósito de eliminar unas finas rayas de color azul que aparecían en las imágenes que no eran por representar una patología sino por cómo estaban armadas las imágenes. Aunque este problema se asociaba principalmente a las patologías de color azul, tras varias pruebas se comprobó que también llegaba a afectar al resto de colores cuando no se aplicaba este primer filtrado. Posteriormente, se incorporó una función de relleno de huecos en el polígono que se puede encontrar en el Anexo II. En un principio esta técnica estaba pensada únicamente para las humedades, ya que sus patrones incluían tramados con múltiples discontinuidades, pero de nuevo, tras varias pruebas, se observó que los grafitis también presentaban huecos en el interior de los polígonos cuando se generaba la máscara sin emplear esta función antes. Aunque para las máscaras rosas bastó con un *kernel* de tamaño (30,30) para las operaciones de dilatación, en el caso de las azules se requirió de uno de (50,50) puesto que los huecos del patrón no terminaban de rellenarse adecuadamente.

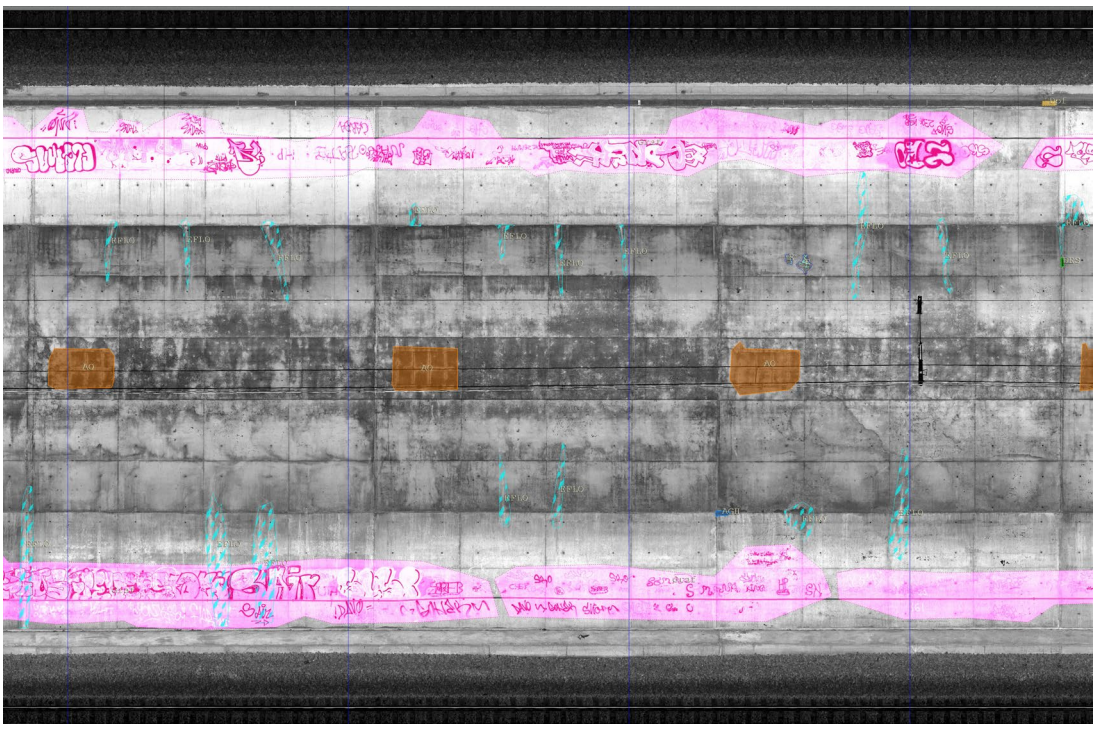


Figura 5. Ejemplo de imagen de ficheros PAT con patologías rosas

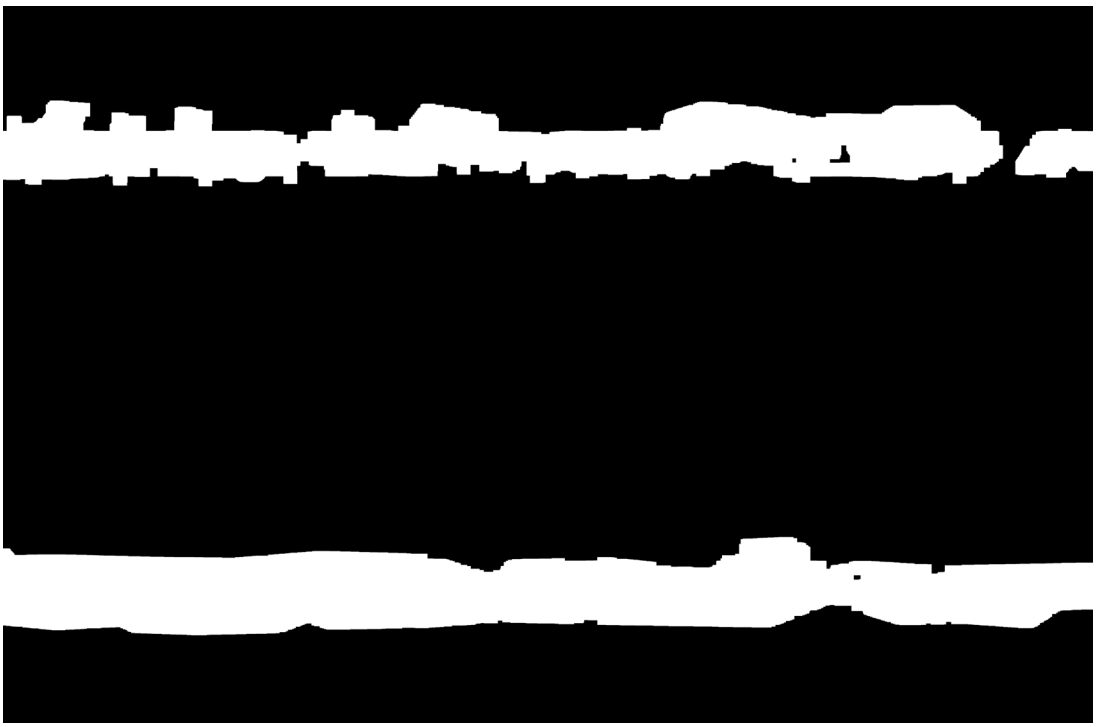


Figura 6. Máscara rosa correspondiente a la Figura 6.

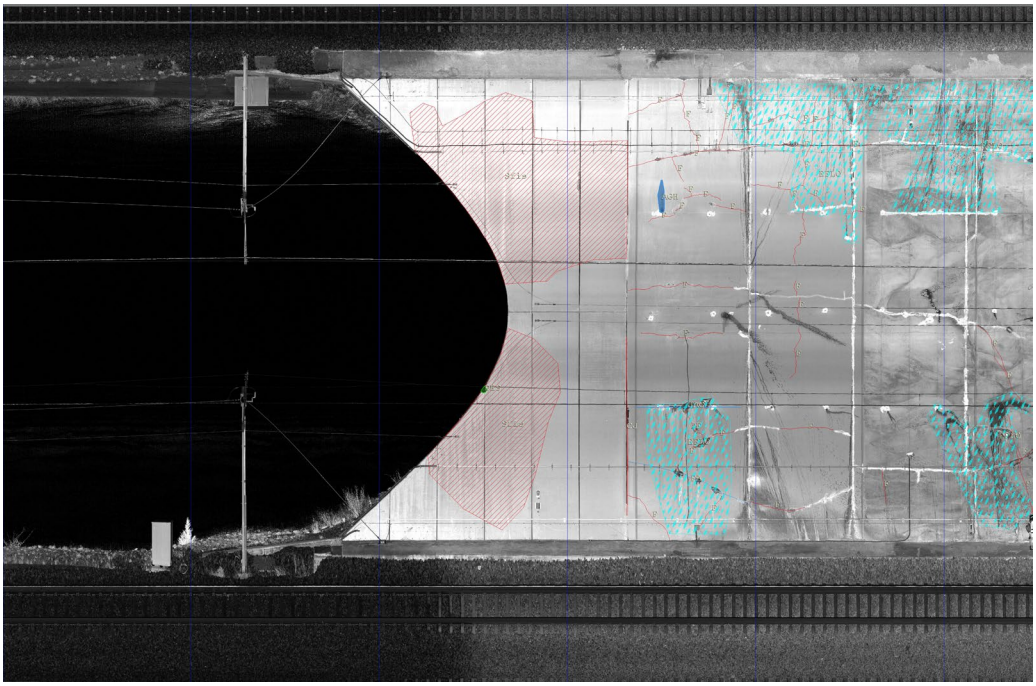


Figura 7. Imagen de ficheros PAT con patologías azules

Figura 8. Máscara azul correspondiente a la Figura 8

Las figuras mostradas anteriormente muestran ejemplos de los pares imagen/máscara para cada uno de los grupos de patologías que se tratan en este trabajo. En las imágenes originales puede observarse con claridad las líneas finas azules que se deciden eliminar antes de generar las máscaras mediante una operación de erosión con un *kernel* de tamaño reducido (5,5). Mientras que las máscaras asociadas a las patologías amarillos sí se ajustan prácticamente de manera exacta a los polígonos marcados en su imagen PAT, se puede observar que en el caso de las máscaras rosas y azules la región que engloban es ligeramente mayor a la delimitada en el polígono original. Esto se debe a la función de solidificación que ambas usan y sobre todo, en el caso de las azules, al uso de un *kernel* mayor (50,50). Conforme el tamaño de *kernel* aumenta, más engrosan los contornos, lo cual puede fusionar objetos cercanos o alterar la geometría original del efecto. Este es un compromiso deliberado, optando por una ligera sobreestimación del área patológica a cambio de obtener una máscara robusta y sin discontinuidades.

5.1.2 RECORTE EN 256X256 CON SOLAPAMIENTO

La decisión de generar primero las máscaras sobre las imágenes completas y luego dividir en bloques de 256x256 píxeles por razones prácticas. Como se mencionó anteriormente, obtener las funciones de generación de máscaras requirió de numerosas pruebas y ajustes de los parámetros (umbrales de color, operaciones morfológicas y su orden, tamaño de *kernel,* relleno de huecos, etc.). De haberse realizado el teselado antes de estas funciones, cada modificación hubiera obligado a regenerar miles de bloques, con un considerable coste de tiempo y de memoria. Por lo contrario, al trabajar sobre las imágenes completas, la verificación resultaba más eficaz ya que bastaba con superponer la máscara sobre la imagen original, ajustar los parámetros y volver a ejecutar el proceso.

Para compatibilizar las imágenes de gran formato con la entrada del modelo, se realizó el recorte de éstas en bloques de 256x256 píxeles con un solapamiento del 50% a excepción de los bordes derechos e inferiores donde se utilizó un solapamiento superior para asegurar que las dimensiones de estos últimos recortes fueran las deseadas. Aumentando el

solapamiento en los márgenes se evitó tanto el uso de relleno artificial (*padding*) como la generación de *tiles* incompletos.

El uso del solapamiento aporta una mayor robustez en el entrenamiento ya que al superponerse parcialmente se ve un mismo defecto desde múltiples posiciones y contextos dentro de distintos mosaicos, enriqueciendo la representación aprendida por la red neuronal. Además, sin solapamiento, en defectos muy pequeños o alargados un corte podría fragmentarlos excesivamente, de manera que el solapamiento asegura que al menos uno de los mosaicos captura el defecto íntegramente. No obstante, el solapamiento incrementa el volumen de datos generados, lo cual puede aumentar los costes de almacenamiento y de entrenamiento.

Antes se ha hablado de la sobre expansión de las máscaras como como un compromiso deliberado, si bien esto es cierto, también es importante reconocer qué limitaciones trae consigo y su posible efecto en este trabajo. Por un lado, debido a las operaciones de dilatación implementadas, las máscaras engloban áreas ligeramente más amplias que las marcadas originalmente por la empresa. Sin embargo, aunque el contorno se hubiese ceñido al proporcionado en los polígonos de los ficheros PAT, puesto que se ha decidido generar una máscara continua de manera que el área englobada por el contorno quede relleno y sea una figura continua, se da el caso tanto en las humedades como en los grafitis en que quedan “pintadas” partes que realmente no son un defecto como tal. Aunque este problema es bastante irrelevante cuando se considera la imagen completa, al recortar en mosaicos de una dimensión bastante menor, dicha operación equivale a realizar un *zoom* local. Esto puede provocar que ciertos bloques contengan fragmentos de máscara, por lo tanto esa imagen de 256x256 contendría algo de blanco indicando defecto, que no corresponden exactamente a un defecto real. Esta situación podría darse por ejemplo en los huecos entre letras de un grafiti o un borde expandido que invade la zona contigua.

Si bien es verdad que lo mencionado podría perjudicar de cierto modo al entrenamiento mediante la introducción de cierto nivel de falsos positivos, aún así se consideró un compromiso aceptable incluso con estas pequeñas imprecisiones puesto que

las máscaras ofrecen una localización bastante aproximada. El objetivo de este trabajo no es la automatización total, sino la aceleración del proceso manual de inspección. Idealmente, con las mejoras necesarias sí se buscaría la automatización total de una forma fidedigna, mientras tanto se aceleran estas tareas tediosas indicando una región de interés que permite al supervisor técnico acudir directamente a ella y examinarla en detalle.

5.1.3 DATA AUGMENTATION

El uso de redes neuronales como la U-Net requiere conjuntos de datos suficientemente variados como para evitar el sobreajuste y conseguir que el modelo generalice bien frente a nuevas muestras. Por esta razón, se hizo uso de técnicas de *data augmentation* ya que permiten abordar estas limitaciones incrementando la variabilidad del conjunto de datos sin necesidad de recopilar nuevas imágenes, lo cual se traduce en un modelo más robusto y adaptable a condiciones reales de inspección.

En primer lugar, se aplicaron transformaciones individuales como rotaciones, volteos verticales y horizontales y desenfoques sobre un porcentaje de las imágenes. En el caso de las transformaciones geométricas, puesto que alteran la localización espacial de los defectos, fue necesario aplicar las transformaciones tanto a las imágenes como a sus máscaras asociadas. El único caso donde se mantenía la máscara inalterada es al aplicar la transformación de desenfoque. Ésta se aplica únicamente sobre la imagen ya que únicamente modifica propiedades visuales y como las máscaras son representaciones binarias de los defectos, cualquier modificación de este tipo hubiera generado incoherencias entre la imagen y su máscara. Posteriormente, se aplicaron transformaciones combinadas, cuyo resultado es la aplicación de dos o más operaciones en secuencia sobre una misma imagen. Estas combinaciones resultan especialmente útiles para reforzar la capacidad del modelo de reconocer defectos en condiciones diversas y desafiantes, asemejándose más a la realidad.

5.2 ENTRENAMIENTO DE LA RED U-NET

Diseñar una U-Net implica programar manualmente cada uno de sus bloques, definir la gestión de dimensiones en las operaciones de convolución, *pooling* y *upsampling*, y asegurar la correcta conexión entre la fase decodificación y la de decodificación. Aparte de laborioso, este proceso aumenta el riesgo de errores de implementación. Es por ello por lo que tomar como base una implementación existente resulta ventajoso, ya que permite apoyarse en un código probado y funcional, además de que acelera el desarrollo. Se ha tomado como referencia el código disponible en un repositorio público de Github (Brendan0403, 2018). Dicho repositorio proporciona una implementación funcional de la arquitectura U-Net aplicada a imágenes médicas. A partir de ese punto de partida, se adaptó el código para encajar con las necesidades del trabajo. Principalmente, se modificó la red para trabajar con imágenes de 256x256, ya que la original trabajaba con imágenes de 128x128. Una vez definida la red, se configuraron dos funciones de perdida: *dice los*s, orientada a maximizar la superposición entre predicción y máscara y la *combined dice + binary crossentropy*, que añadía estabilidad durante la optimización.

El primer entrenamiento se llevó a cabo con un fichero pequeño de imágenes de grafitis, elegido precisamente por su tamaño reducido ya que era de los pocos conjuntos que la RAM podía soportar en su totalidad. Los resultados iniciales parecían extremadamente prometedores, con métricas de validación tan elevadas que resultaban extrañas. Tras analizar los resultados, pudo observarse que había un gran desbalanceo de clases y que la mayoría de las imágenes no contenían ningún defecto y, por lo tanto, la mayoría de máscaras asociadas eran bloques completamente negros. Por ende, no es que el modelo inicial fuera verdaderamente prometedor, sino que había aprendido a detectar “no defecto” de manera muy precisa, sin suponer una segmentación útil de grafitis.

Esto último pudo verificarse rápidamente revisando los *true positives* sobre varias predicciones, evidenciando que el modelo apenas segmentaba grafitis. Para abordar esta limitación, se diseñó un conjunto de entrenamiento alternativo de tamaño N, de tal forma que las imágenes con defecto del fichero compusieran el 80% de este conjunto y el 20%

fueran imágenes sin defecto obtenidas del fichero de manera aleatoria. Debido a que el número absoluto de imágenes con defecto del fichero era bastante reducido, se aplicaron técnicas de *data augmentation* para compensar esta carencia y evitar que la red memorizara únicamente los ejemplos disponibles. Aunque de esta forma se pudieron aumentar artificialmente el número de muestras positivas y aportar mayor variabilidad, el tamaño final de N continuó siendo relativamente pequeño. Pese a esta restricción, se obtuvieron unos resultados razonablemente buenos demostrando que el modelo era capaz de extraer patrones relevantes para la tarea.

Con el objetivo de comprobar el efecto del tamaño total del conjunto en el aprendizaje independientemente de la proporción de imágenes con y sin defecto el modelo mejoraba su capacidad de aprendizaje, se decidió pasar a entrenar con humedades. Se fijó un tamaño de conjunto de 6000 imágenes, el cual representaba el máximo volumen que podía manejar la RAM del equipo sin provocar fallos durante la carga de datos. Únicamente se consideró el tamaño definitivo del conjunto, sin garantizar un porcentaje determinado de imágenes con defecto. De manera consistente con los experimentos previos, se utilizó una U-Net clásica junto con la función de pérdida *dice_bce_combined.*

Considerando que, según la literatura la Attention U-Net resulta atractiva en escenarios con un fuerte desbalance de clases, se decidió probar el mismo conjunto desbalanceado de 6000 imágenes para observar cómo se comportaba la Attention U-Net. Tras varias pruebas donde se observó una finalización prematura del entrenamiento, debido al *early stopping* que decidía en base a la función de pérdida en validación a pesar de que las métricas de entrenamiento parecían mejorar consistentemente, se decidió no emplear un conjunto de validación, a diferencia del resto de casos desarrollados en este trabajo.

Posteriormente, se decidió recrear el modelo que había proporcionado resultados decentes para detectar grafitis y aplicarlo para detectar humedades. De nuevo se utilizó una U-Net clásica con funciones de pérdida *dice* y *dice_bce_combined*. Al igual que en el caso de aplicación en los grafitis, se obtuvieron resultados optimistas. Sin embargo, al visualizar algunas de las predicciones realizadas sobre el conjunto de validación frente a las máscaras

correspondientes que se usaron para entrenar al modelo, las predicciones no eran muy similares a la máscara real. Es por ello por lo que se decidió modificar el valor de umbral usado al realizar las predicciones y tras disminuirlo, el coeficiente *dice* mejoró notablemente y las predicciones resultaron bastante parecidas a las máscaras reales. Este cambio de umbral se aplicó sobre el último modelo de grafitis y también se observó un cambio favorable en su coeficiente *dice.*

Capítulo 6. RESULTADOS Y DISCUSIÓN

6.1 GRAFITIS

Se utilizó un pequeño conjunto de 1547 imágenes alcanza un coeficiente Dice de 0.8408 en entrenamiento, 0.7382 en validación y 0.8290 en test. Pese a la limitación de datos, aun habiendo aumentado artificialmente el tamaño del conjunto de grafitis, las métricas reflejan que el modelo fue capaz de aprender patrones útiles para segmentar grafitis y generalizar razonablemente bien hacia datos no vistos. la variabilidad introducida tras aplicar técnicas de *data augmentation* (rotaciones, espejados o cambios de escala) favoreció que aprendiera a reconocer grafitis bajo distintas orientaciones y tamaños, centrándose en patrones invariantes más representativos. Este efecto explica que, a pesar de contar con un conjunto reducido de imágenes, se alcanzaran resultados consistentes tanto en validación como en prueba.

Las gráficas de evolución (Figuras 10 y 11) muestran un coeficiente Dice que asciende de forma continua tanto en entrenamiento hasta alcanzar valores en torno a 0.84 en entrenamiento y 0.74 en validación. La diferencia entre entrenamiento y validación es esperable y, dado que el rendimiento de prueba (0.8290) se mantiene en niveles cercano al de entrenamiento, no constituye evidencia de sobreajuste. En cuanto a la función de pérdida, mientras que en entrenamiento decrece sostenidamente hasta valores bajos, en validación se aprecian oscilaciones marcadas, principalmente durante las primeras épocas, antes de estabilizarse hacia la mitad del proceso.

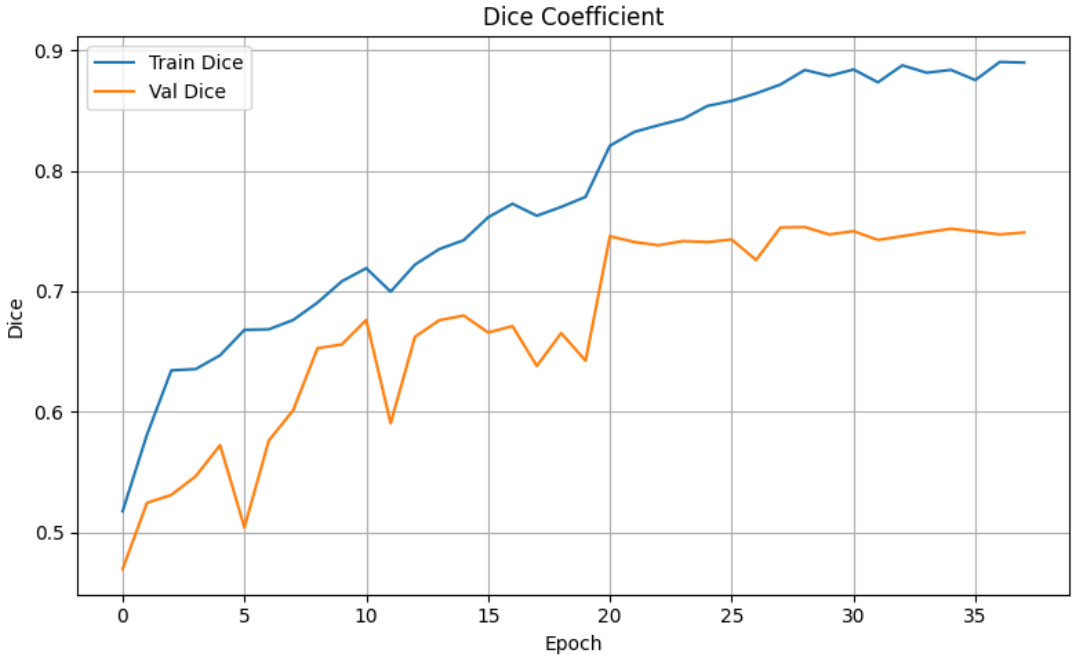


Figura 9. Coeficiente Dice de los conjuntos de entrenamiento y validación a lo largo de las épocas

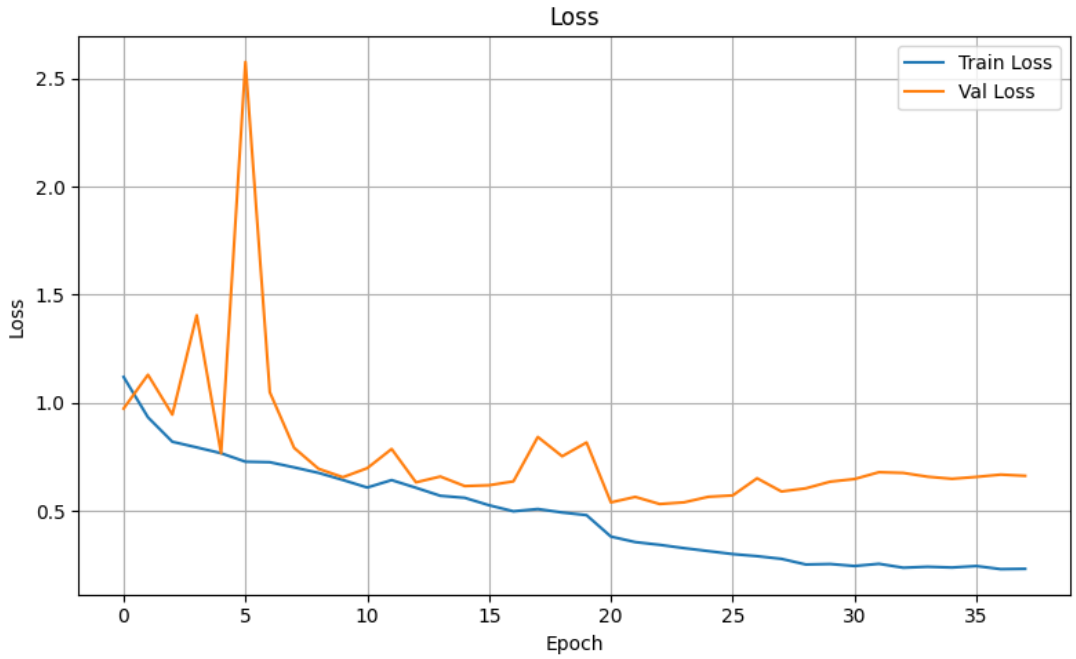


Figura 10. Función de pérdida Dice de los conjuntos de entrenamiento y validación a lo largo de las épocas

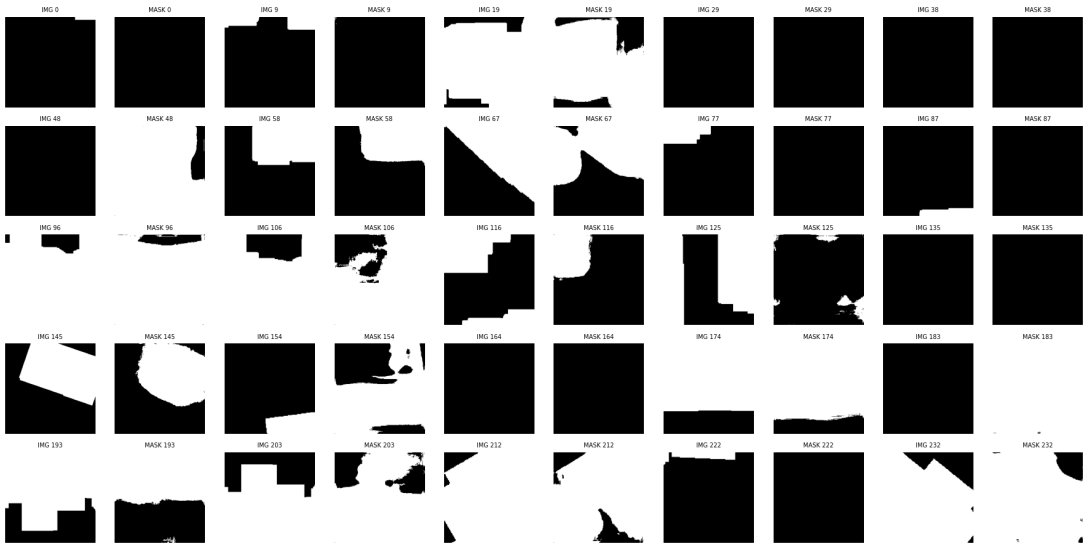


Figura 11. Comparación de máscaras reales y predicciones del modelo en el conjunto de prueba

La Figura 12 compara visualmente las máscaras reales y las predicciones obtenidas en el conjunto de prueba, pudiendo apreciar de manera cualitativa el comportamiento del modelo. Aunque es cierto que las segmentaciones presentan limitaciones notables, considerando la escasez de datos posibilidades reales y la gran variabilidad intrínseca del grafiti (colores, tipografías, texturas), el sistema consigue identificar la presencia del grafiti y reproducir su localización general. El modelo evidencia que ha captado patrones representativos del defecto, manteniendo una capacidad razonable de generalización para ubicar grafitis en imágenes no observadas previamente.

Una posible explicación de porqué, tras múltiples pruebas no se obtuvo una mejoría puede residir en cómo la empresa definió las regiones de defecto inicialmente en los ficheros PAT. La figura 3 muestra como la anotación rosa no delimita únicamente el área ocupada por el grafiti, sino que engloba una región que contiene una cantidad notable de pared limpia que tras generar la máscara (como detecta rosa) etiqueta la zona entera como grafiti. Al entrenar sobre este tipo de anotaciones, el modelo aprender a segmentar no solo el grafiti sino también parte del fondo, lo cual reduce precisión en los contornos y explica la obtención de métricas más irregulares. La forma de verificarlo sería reducir la resolución original de

las imágenes de manera que cada recorte de 256x256 abarque una región más amplia del túnel. De esta forma, el modelo recibiría, además del detalle local del grafiti, parte del contexto espacial (e.g. pared circundante o cambios de textura), lo cual le permitiría reducir la confusión con zonas de pared sin grafiti y distinguir con mayor precisión los límites reales del grafiti.

6.2 HUMEDADES

6.2.1 CONJUNTO DESBALANCEADO

6.2.1.1 U-Net clásica

El primer modelo de humedades se entrenó con 6000 imágenes seleccionadas de manera aleatoria sin garantizar un porcentaje de imágenes con defectos. El desempeño fue considerablemente inferior, con los siguientes coeficientes Dice: 0.3594 en entrenamiento, 0.3312 en validación y 0.6726 en prueba. Estas métricas bajas reflejan un aprendizaje deficiente por parte de la red, donde la presencia excesiva de imágenes sin defecto en el conjunto de entrenamiento provocó que el modelo no aprendiera adecuadamente los patrones positivos. El hecho de que en el conjunto de prueba se alcanzara un coeficiente Dice superior no significa una mejoría del modelo, sino que, por la composición del conjunto de prueba, sus condiciones resultaron más favorables de forma coyuntural.

En las gráficas de entrenamiento y validación (Figuras 13 y 14) puede apreciarse de forma clara su bajo rendimiento, donde el coeficiente Dice apenas supera 0.35 y ni el entrenamiento, ni mucho menos la validación, alcanzan una cierta estabilidad ni una convergencia clara. Aunque podría argumentarse que un mayor número de épocas podría mejorar el ajuste, dado que se aplicó *early stopping* y que las curvas no muestran una tendencia clara hacia la mejora, es poco probable que un entrenamiento más prolongado hubiese supuesto una diferencia significativa en el rendimiento final.

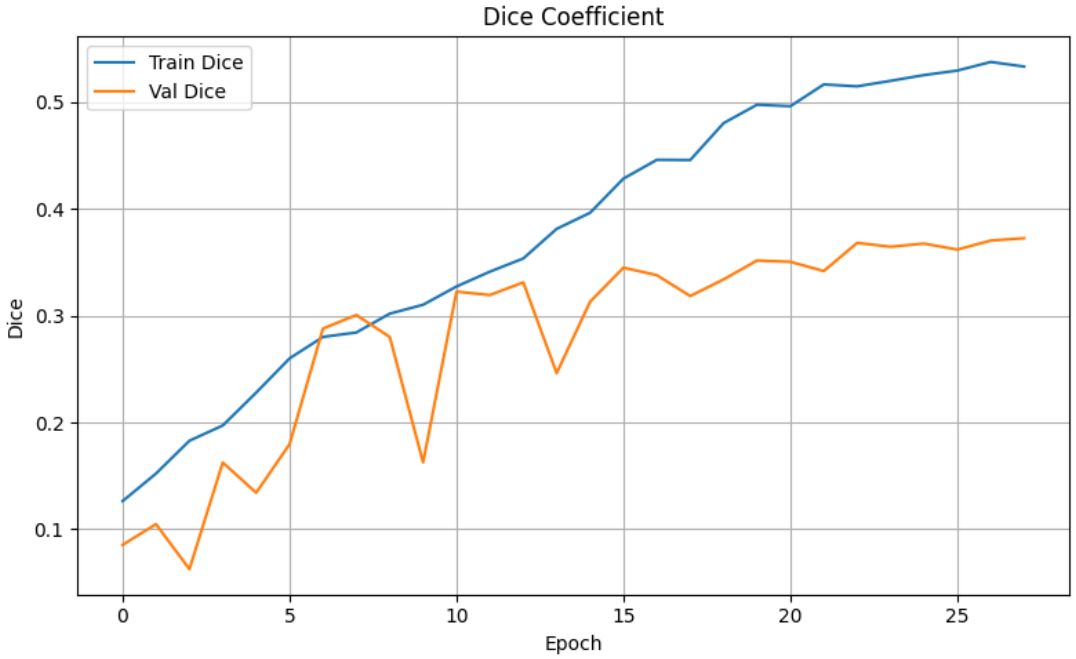


Figura 12. Coeficiente Dice de los conjuntos de entrenamiento y validación a lo largo de las épocas

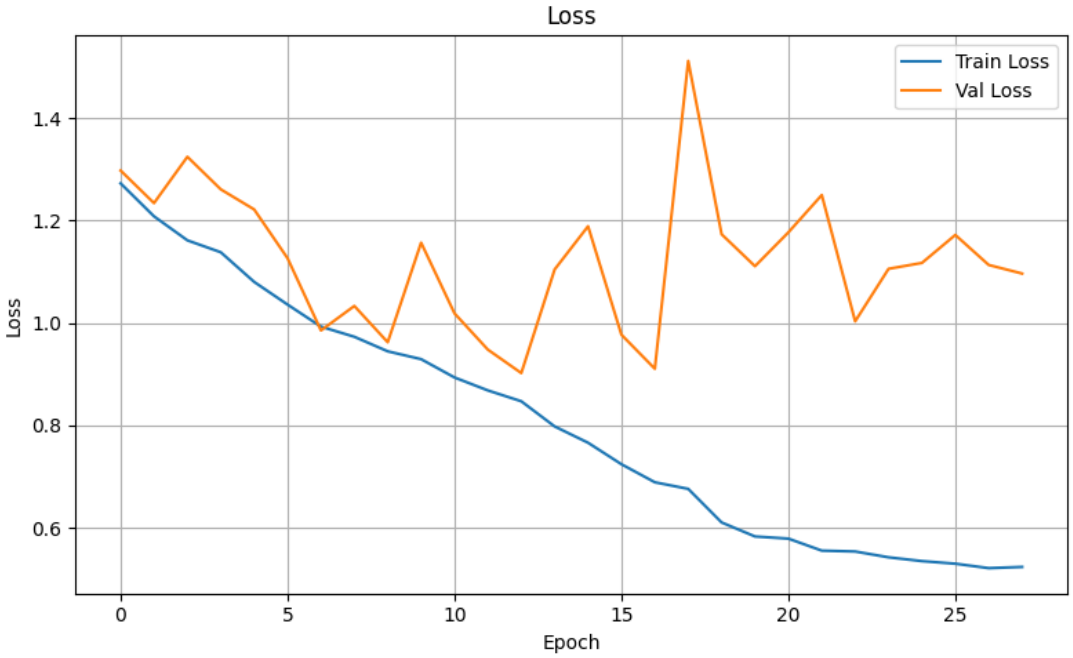


Figura 13. Función de pérdida Dice de los conjuntos de entrenamiento y validación a lo largo de las épocas

6.2.1.2 Attention U-Net

En el entrenamiento de la Attention U-Net sobre el conjunto desbalanceado de 6000 imágenes se obtuvieron resultados superiores a los alcanzados con la U-Net clásica. El modelo alcanzó un coeficiente dice de 0,8530 en entrenamiento y de 0,7673 en el conjunto de prueba, demostrando una mejora en la segmentación de humedades frente al modelo base. Como evidencian los resultados, la incorporación de mecanismos de atención permitió focalizar el aprendizaje en las regiones minoritarias, mitigando los efectos del balance de clases y ofreciendo segmentaciones visualmente más consistentes que los de la U-Net clásica, como puede apreciarse en la siguiente imagen:

Figura 14. Comparación de máscaras reales y predicciones del modelo en el conjunto de prueba

6.2.2 CONJUNTO BALANCEADO

Tras identificar el problema del modelo anterior, se construyó un nuevo conjunto de 5248 imágenes en el que se garantizó un 80% de ejemplos con defecto y un 20% sin defecto. Bajo esta configuración, los resultados mejoraron significativamente, alcanzando un coeficiente Dice de 0.98 en entrenamiento, 0.94 en validación y 0.89 en test. A pesar de estar balanceado

de la misma manera que el conjunto usado en el modelo de grafitis (con un 80% de imágenes con defecto), el cual obtuvo resultados más modestos, al contener más cantidad de imágenes con defecto, su tamaño de N fue superior al tamaño de N usado en el caso de los grafitis. Esto sugiere que disponer de un conjunto de datos más amplio resulta determinante para el aprendizaje, logrando que la red fuera capaz de capturar de forma robusta las características de las humedades y generando predicciones más estables y coherentes con las máscaras de referencia. Por otro lado, los resultados muestran que a pesar de tener un menor número de imágenes que el conjunto desbalanceado, el desempeño es notablemente superior, evidenciando que la proporción adecuada de clases es esencial para que el modelo aprenda a identificar correctamente las humedades.

La curva del coeficiente Dice en la Figura 16 muestra un crecimiento progresivo y sostenido en entrenamiento, mientras que en validación el comportamiento es más inestable durante las primeras épocas donde la curva oscila pronunciadamente, reflejando la dificultad inicial del modelo para generalizar a ejemplos no vistos. A partir de la época 30, ambas curvas comienzan a estabilizarse, confirmando que el modelo logra consolidar un aprendizaje robusto. Por su parte, en la Figura 17, la curva de función de pérdida muestra una disminución pronunciada en el conjunto de entrenamiento que se aproxima a cero en las épocas finales. La pérdida de validación desciende de forma irregular hasta que eventualmente se estabiliza en niveles bajos a partir de la época 30 también.

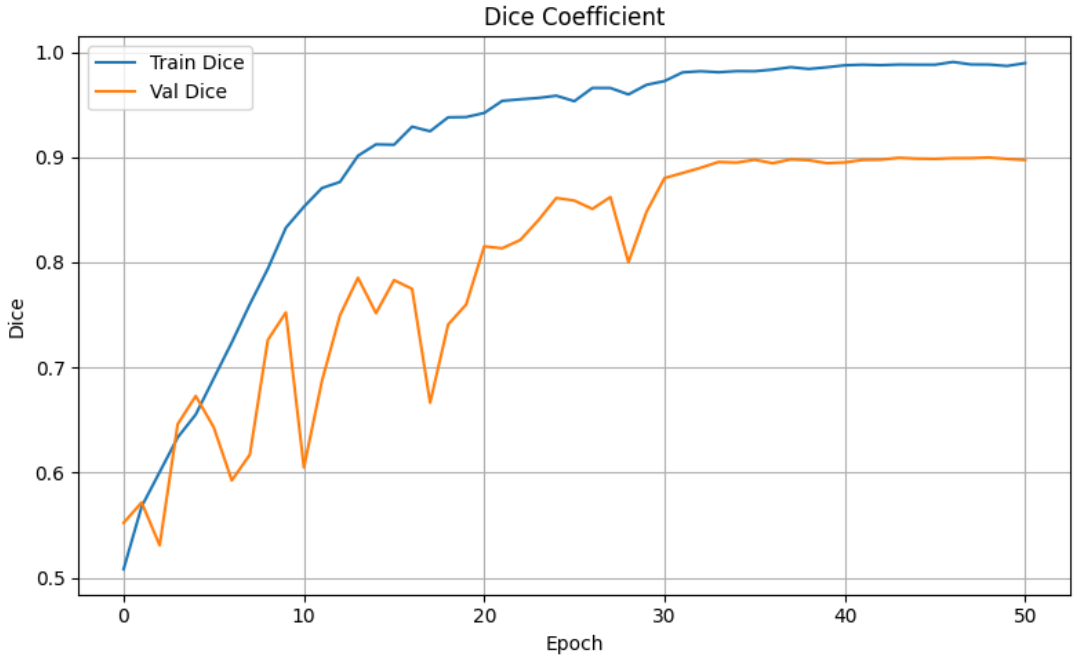


Figura 15. Coeficiente Dice de los conjuntos de entrenamiento y validación a lo largo de las épocas

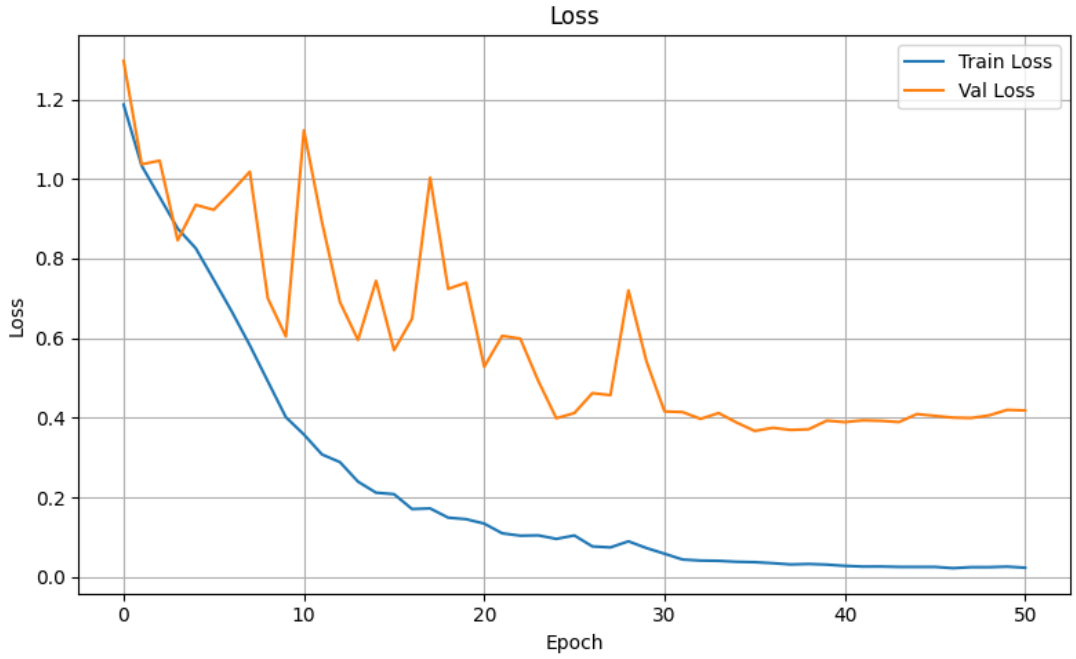


Figura 16. Función de pérdida Dice de los conjuntos de entrenamiento y validación a lo largo de las épocas

En la Figura 16, se observa que el conjunto de validación se estabiliza en torno a 0.90, este valor refleja el cálculo directo sobre las probabilidades de salida del modelo con el umbral estándar de 0.5. Sin embargo, al ajustar el umbral de decisión a un valor inferior de 0.3, se obtuvo un dice de 0.94 entre las predicciones de validación y las máscaras reales. Este resultado evidencia que la correcta calibración del umbral de predicción puede optimizar la segmentación en función del tipo de defecto. En este caso, el modelo tendía a ser conservador en sus predicciones, por lo que reducir el umbral aumentó su sensibilidad de manera que se capturaran mejor las regiones de humedad. A continuación, se muestran ejemplos de las máscaras predichas por el modelo sobre el conjunto de prueba:

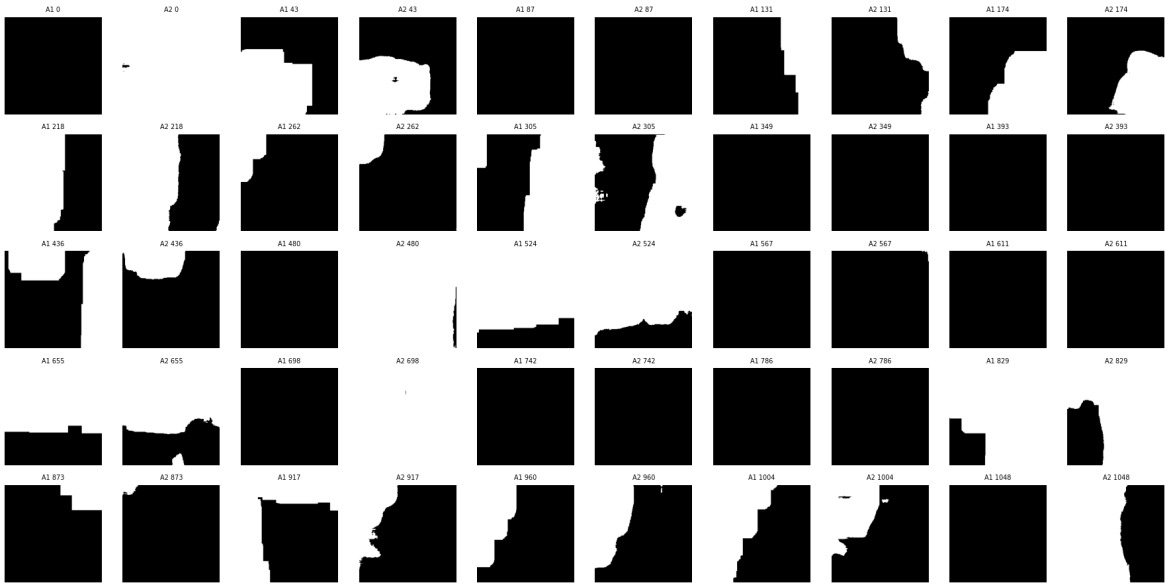


Figura 17. Comparación de máscaras reales y predicciones del modelo en el conjunto de prueba

En la Figura 18 se puede observar una alta correspondencia entre las máscaras reales y las predicciones del modelo en el conjunto de prueba. En la mayoría de los casos, el modelo consigue identificar la presencia de humedad y reproducir su extensión. No obstante, se aprecian algunas limitaciones como los bordes que aparecen sobre rellenados o ligeramente desplazados, mientras que en otros defectos más pequeños o con formas irregulares el modelo simplifica la geometría real. A pesar de estas limitaciones, la localización sigue

siendo bastante precisa y la segmentación resultante es operativamente útil ya que dirige la atención hacia las zonas efectivamente afectadas.

Los resultados obtenidos pueden verse resumidos en la siguiente tabla:

Defecto	**Número de imágenes**	**Proporción de clases**	**Dice entrenamiento**	**Dice validación**	**Dice prueba**	**Tipo de U-Net**
Grafitis	1547	80% con defecto	0.8404	0.7382	0.8290	Clásica
Humedades	6000	Aleatoria	0.3594	0.3312	0.6726	Clásica
Humedades	6000	Aleatoria	0.8530	-	0.7673	Attention
Humedades	5248	80% con defecto	0.9785	0.9400	0.8900	Clásica

Tabla 2. Comparación de los resultados obtenidos

CONCLUSIONES Y TRABAJOS FUTUROS

La viabilidad de aplicar arquitecturas de segmentación basadas en U-Net para la detección automática de defectos en túneles ferroviarios queda confirmada mediante los resultados obtenidos. El sistema ha demostrado rendimientos sólidos, especialmente en el caso de las humedades, reflejando su capacidad de aprendizaje y generalización. No obstante, también se han identificado limitaciones relevantes. Por un lado, la elevada variabilidad en colores, tonos y patrones de los defectos (como se aprecia en la Figura 5), así como la forma en que se definen los polígonos de colores en los ficheros PAT, donde abarcan áreas más extensas que el defecto real, introduce ruido que afecta de manera significativa a la precisión de las segmentaciones.

A estas limitaciones asociadas a los datos se suman las restricciones computacionales. Cada entrenamiento completo requería entre 12 y 24 horas, suponiendo una barrera significativa para realizar ajustes de parámetros o probar variantes de manera ágil. Cualquier ajuste experimental se convertía en un proceso lento, por lo que para este tipo de trabajos resulta fundamental contar con hardware que permita entrenar y validar en tiempos razonables. La falta de memoria RAM y la dependencia de recursos compartidos en Colab redujeron la posibilidad de exploras alternativas con mayor profundidad, como muestrear las imágenes originales para comprobar la entrada de defectos más grandes en los *patches* de 256x256, lo cual fue una idea planteada pero eventualmente inviable. Bajo estas condiciones, los resultados obtenidos permiten extraer varias conclusiones relevantes:

1. El sistema ha demostrado ser capaz de aprender y generalizar patrones útiles, cumpliendo con el objetivo principal de demostrar la viabilidad de un enfoque basado en la segmentación de imágenes con U-Net para la detección automática de defectos en túneles ferroviarios.
2. En el caso de los grafitis, con un conjunto reducido de 1547 imágenes, el modelo alcanzó métricas relativamente buenas, suponiendo un cumplimiento parcial de los objetivos plantados.

3. Cuando se trabajó con un conjunto más amplio sin balanceo de clases se observó un extremo deterioro en la capacidad de aprendizaje del modelo, fracasando en la detección de patrones positivos.
4. El uso de la Attention U-Net ante un conjunto desbalanceado permitió mejorar los resultados respecto a la U-Net clásica
5. En el caso de las humedades con 5248 imágenes, el rendimiento fue considerablemente superior, confirmando un cumplimiento total de los objetivos y demostrando que, con un conjunto más amplio y balanceado, la red es capaz de generar segmentaciones estables y coherentes.
6. La calibración del umbral de predicción resultó ser un factor clave para optimizar el rendimiento, logrando incrementos notables en las métricas.
7. El trabajo desarrollado valida la aplicabilidad del enfoque propuesto y sienta una base sólida para futuros desarrollos, pudiendo concluirse que el trabajo ha sido adecuado, correcto y que cumple con los objetivos establecidos.

6.3 ESTIMACIÓN DEL VOLUMEN DE DATOS

Tomando como referencia el fichero utilizado para entrenar el modelo de humedades, un fichero de 10 páginas que generó 26680 recortes de 256x256 (ocupando 2.83GB en disco), se obtiene una media de 2668 imágenes por página. Si se extrapola este valor al total de los 118 ficheros proporcionados por la empresa, con una media estimada de 60 páginas por fichero (considerando que este número varía de 5 a 200 páginas según el fichero), el volumen potencial alcanzaría las 7000 imágenes (hay una imagen por página). De estas imágenes se obtendrían entre 18 y 19 millones de imágenes de 256x256, lo cual quiere decir que, en términos de almacenamiento, este conjunto supondría unos 2TB de datos únicamente en imágenes RGB, sin tener en cuenta las tres máscaras asociadas (rosa, azul, amarillo) a cada recorte de 256x256.

En la práctica, la memoria RAM impedía cargar más de 6000 imágenes simultáneamente, razón por la cual se tuvo que trabajar con subconjuntos reducidos. Aún

cuando se intentó procesar los datos por lotes y posteriormente concatenarlos para el entrenamiento, la RAM inevitablemente colapsaba, haciendo inviable el aprovechar un volumen mayor de información. A lo largo del proyecto se emplearon unas 18795 imágenes, representando apenas un 0.1% del *dataset* disponible. Con una fracción tan reducida, haber obtenido resultados decentes refuerza la solidez del enfoque y evidencia el potencial de mejora que podría lograrse si se dispusiera de los recursos computacionales necesarios para aprovechar la totalidad de datos proporcionados por la empresa.

6.4 *TRABAJOS FUTUROS*

El presente trabajo ha permitido validar la viabilidad de emplear la U-Net y la Attention U-Net para la detección automática de defectos en túneles ferroviarios. Sin embargo, el margen de mejora identificado es amplio y es por ello por lo que, de cara a futuras investigaciones, se plantean varias líneas de desarrollo. Una primera propuesta sería el aprovechamiento completo del *dataset* proporcionado por Axil Ingeniería y Servicios. Contar con un hardware de mayor capacidad de cómputo permitiría entrenar con la totalidad del conjunto de datos, reduciendo el sesgo de muestreo y mejorando la capacidad de generalización del modelo. Además, supondría hacer justicia a la intención inicial de este trabajo que precisamente pretendía evaluar el sistema en toda su magnitud. Esta vía podría materializarse, por ejemplo, mediante el acceso administrado a superordenadores y clústeres de cálculo disponibles en diversos centros universitarios o de investigación.

En cuanto a las arquitecturas empleadas, se implementaron la U-Net clásica y la Attention U-Net. No obstante, la literatura ofrece otras variantes prometedoras que no se han explorado en este trabajo. En este contexto, algunas resultan especialmente relevantes ya que los defectos con los que se ha lidiado presentan patrones dispersos, tonalidades muy variadas y tamaños heterogéneos. Entre ellas, se encuentra la U-Net++ que rediseña la fase de decodificación mediante conexiones densas para mejorar la segmentación de estructuras con bordes complejos. Por otra parte, existen variantes que incorporan mecanismos de *Transfromers*, lo cual les permite capturar dependencias de largo alcance en la imagen y aprovechar mejor el contexto global. La TransUNet combina la arquitectura de U-Net con

bloques de autoatención y la Swin-UNet emplea *SwinTransformers* (Ashish bisht, 2024) con ventanas jerárquicas deslizantes para procesar a múltiples escalas la información. Aunque su entrenamiento implicaría un mayor coste computacional, estas aproximaciones podrían aportar mejoras significativas en la precisión y robustez de los resultados.

Otra línea de investigación considera el uso de arquitecturas de detección de objetos. En múltiples de los estudios revisado en el capítulo de estado de la cuestión se emplearon CNNs convencionales en lugar de segmentación semántica. Los modelos de detección trabajan con cajas delimitadoras, a diferencia de las redes de segmentación que generan máscaras a nivel de píxel, lo que reduce la complejidad del proceso de anotación y acelera el entrenamiento. Además, en contextos donde la prioridad es identificar la presencia y localización aproximada de un defecto, más que su contorno exacto, estas redes pueden resultar más eficientes. Por ello, podrían ofrecer un mejor equilibrio entre precisión, coste computacional y viabilidad práctica.

Finalmente, sería recomendable, en colaboración con la empresa, redefinir el proceso de etiquetado con polígonos que se ciñan al contorno real de los defectos y limitando el uso de colores. Un etiquetado más homogéneo y ajustado facilitaría la extracción automática de las máscaras, reduciría el ruido en el entrenamiento y permitiría que los modelos aprendieran patrones más consistentes, mejorando así la fiabilidad de las segmentaciones.

Capítulo 7. BIBLIOGRAFÍA

Abdel-Qader, I., Abudayyeh, O., & Kelly, M. E. (2003). Analysis of Edge-Detection Techniques for Crack Identification in Bridges. *Journal of Computing in Civil Engineering*, *17*(4), 255–263. https://doi.org/10.1061/(ASCE)0887-3801(2003)17:4(255)

Abdel-Qader, I., Pashaie-Rad, S., Abudayyeh, O., & Yehia, S. (2006). PCA-Based algorithm for unsupervised bridge crack detection. *Advances in Engineering Software*, *37*(12), 771–778. https://doi.org/10.1016/j.advengsoft.2006.06.002

Alpkokin, P. (2012). Historical and critical review of spatial and transport planning in the Netherlands. *Land Use Policy*, *29*(3), 536–547. https://doi.org/10.1016/j.landusepol.2011.09.007

Analytics Vidhya. (2025, May 1). *Image Segmentation with U-Net*. https://www.analyticsvidhya.com/blog/2022/10/image-segmentation-with-u-net/

Ashish bisht. (2024, February 17). *Swin-Transformer-based Unet architecture for semantic segmentation with Pytorch code*. https://medium.com/@ashishbisht0307/swin-transformer-based-unet-architecture-for-semantic-segmentation-with-pytorch-code-91e779334e8e

Brendan0403. (2018, September 9). *Bones Segmentation*. https://github.com/brendan0403/BonesSegmentation/commits?author=brendan0403

Chen, L.-P. (2019). Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar: Foundations of machine learning, second edition. *Statistical Papers*, *60*(5), 1793–1795. https://doi.org/10.1007/s00362-019-01124-9

Choudhary, G. K., & Dey, S. (2012). Crack detection in concrete surfaces using image processing, fuzzy logic, and neural networks. *2012 IEEE 5th International Conference*

on Advanced Computational Intelligence, ICACI 2012, 404–411. https://doi.org/10.1109/ICACI.2012.6463195

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., & Ronneberger, O. (2016). *3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation*. http://arxiv.org/abs/1606.06650

Craig Hale. (2025, August 20). *Python isn't dead- despite funding cuts, programming language powers on*. https://www.techradar.com/pro/python-isnt-dead-despite-funding-cuts-programming-language-powers-on

Flor, A. (2019, March 25). *Surface Crack DEtection*. Medium. https://arthurflor23.medium.com/surface-crack-detection-92fb27bf0f18

Fogaça, J., Brandão, T., & Ferreira, J. C. (2023). Deep Learning-Based Graffiti Detection: A Study Using Images from the Streets of Lisbon. *Applied Sciences (Switzerland)*, *13*(4). https://doi.org/10.3390/app13042249

Ibtehaz, N., & Rahman, M. S. (2019). *MultiResUNet : Rethinking the U-Net Architecture for Multimodal Biomedical Image Segmentation*. https://doi.org/10.1016/j.neunet.2019.08.025

Jadon, S. (2020). *A survey of loss functions for semantic segmentation*. https://doi.org/10.1109/CIBCB48159.2020.9277638

Kasraian, D., Maat, K., & van Wee, B. (2016). Development of rail infrastructure and its impact on urbanization in the Randstad, the Netherlands. *Journal of Transport and Land Use*, *9*(1), 151–170. https://doi.org/10.5198/jtlu.2015.665

Kerem Turgutlu. (2018, April 20). *Semantic Segmentation- U-Net*. https://medium.com/@keremturgutlu/semantic-segmentation-u-net-part-1-d8d6f6005066

Keylabs. (2024, September 2). *Types of Classification Models: Binary, Multiclass and Multilabel*. https://keylabs.ai/blog/types-of-classification-models-binary-multiclass-and-multilabel/

Müller, D., Soto-Rey, I., & Kramer, F. (2022). *TOWARDS A GUIDELINE FOR EVALUATION METRICS IN MEDICAL IMAGE SEGMENTATION*.

Navab, N., Hornegger, J., Wells, W. M., & Frangi, A. F. (2015). LNCS 9351 - Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. In *Proceedings, Part III Medical Image Computing and Computer-Assisted Intervention-MICCAI*. http://www.springer.com/series/7412

Oktay, O., Schlemper, J., Folgoc, L. Le, Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N. Y., Kainz, B., Glocker, B., & Rueckert, D. (2018). *Attention U-Net: Learning Where to Look for the Pancreas*. http://arxiv.org/abs/1804.03999

Qu, Z., Ju, F.-R., Guo, Y., Bai, L., & Chen, K. (2018). Concrete surface crack detection with the improved pre-extraction and the second percolation processing methods. *PLOS ONE*, *13*(7), e0201109. https://doi.org/10.1371/journal.pone.0201109

Robin Vinod. (2020, May 7). *Dealing with Class Imbalanced Image Datasets Using the Focal Tversky Loss*. https://medium.com/data-science/dealing-with-class-imbalanced-image-datasets-1cbd17de76b5

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9351*, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Sacchi, C., Regazzoni, C., & Vernazza, G. (2001). A neural network-based image processing system for detection of vandal acts in unmanned railway environments. *Proceedings* -

11th International Conference on Image Analysis and Processing, ICIAP 2001, 529–534. https://doi.org/10.1109/ICIAP.2001.957064

Samsami, R. (2024). A Systematic Review of Automated Construction Inspection and Progress Monitoring (ACIPM): Applications, Challenges, and Future Directions. In *CivilEng* (Vol. 5, Issue 1, pp. 265–287). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/civileng5010014

Silva, W. R. L. da, & Lucena, D. S. de. (2018). Concrete Cracks Detection Based on Deep Learning Image Classification. *The 18th International Conference on Experimental Mechanics*, 489. https://doi.org/10.3390/ICEM18-05387

Stoyanov, D., Taylor, Z., Carneiro, G., Syeda-Mahmood, T., Martel, A., Maier-Hein, L., Tavares, J. M. R. S., Bradley, A., Papa, J. P., Belagiannis, V., Nascimento, J. C., Lu, Z., Conjeti, S., Moradi, M., Greenspan, H., & Madabhushi, A. (Eds.). (2018). *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support* (Vol. 11045). Springer International Publishing. https://doi.org/10.1007/978-3-030-00889-5

Terven, J., Cordova-Esparza, D. M., Romero-González, J. A., Ramírez-Pedraza, A., & Chávez-Urbiola, E. A. (2025). A comprehensive survey of loss functions and metrics in deep learning. *Artificial Intelligence Review*, *58*(7). https://doi.org/10.1007/s10462-025-11198-7

Zhang, Z., Liu, Q., & Wang, Y. (2018). Road Extraction by Deep Residual U-Net. *IEEE Geoscience and Remote Sensing Letters*, *15*(5), 749–753. https://doi.org/10.1109/LGRS.2018.2802944

Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., & Liang, J. (2020). *UNet++: Redesigning Skip Connections to Exploit Multiscale Features in Image Segmentation*. http://arxiv.org/abs/1912.05074

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

En la planificación urbana contemporánea, cada vez son más relevantes aspectos como la mejora de la calidad del aire urbano, la reducción del ruido y la menor ocupación de aspecto vial. Al integrarse fácilmente con redes metropolitanas, el ferrocarril contribuye a descongestionar los accesos urbanos y reducir la necesidad de infraestructuras viales adicionales, trayendo consigo beneficios que no sólo se limitan al plano ambiental, sino que contribuyen al diseño de ciudades más limpias y conectadas. Es por esto que el ferrocarril aparece como una herramienta clave en consecución de los Objetivos de Desarrollo Sostenible (ODS) 9 (Industria, innovación e infraestructuras) y 11 (Ciudades y comunidades sostenibles).

ANEXO II: FUNCIÓN SOLIDIFICACIÓN MÁSCARAS

```
def solidify_mask(mask):

    mask_bin = np.where(mask > 127, 255, 0).astype(np.uint8)


    contours, _ = cv2.findContours(mask_bin.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    solid_mask = np.zeros_like(mask_bin)
    cv2.drawContours(solid_mask, contours, -1, color=255, thickness=cv2.FILLED)

    return solid_mask
```

ANEXO III: GENERACIÓN MÁSCARA AZUL

```
def generate_blue_mask(image_filename, pdf_name, lower_bound, upper_bound):
    input_path, output_path = get_paths(image_filename, pdf_name, "blue")
    image = cv2.imread(input_path)
    if image is None:
        raise FileNotFoundError(f"No se pudo cargar la imagen en {input_path}")

    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv_image, lower_bound, upper_bound)

    # Kernel pequeño para la eliminar las líneas finas
    small_kernel = np.ones((5, 5), np.uint8)
    mask = cv2.erode(mask, small_kernel, iterations=1)

    # Kernel grande para la máscara en sí
    large_kernel = np.ones((50, 50), np.uint8)
    mask = cv2.dilate(mask, large_kernel, iterations=2)

    cv2.imwrite(output_path, mask)
    crop_and_save_masks(output_path, pdf_name, "blue")
    return output_path
```

ANEXO IV: GENERACIÓN MÁSCARA ROSA

```
def generate_pink_mask(image_filename, pdf_name, lower_bound, upper_bound):
    input_path, output_path = get_paths(image_filename, pdf_name, "pink")
    image = cv2.imread(input_path)
    if image is None:
        raise FileNotFoundError(f"No se pudo cargar la imagen en {input_path}")

    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv_image, lower_bound, upper_bound)

    small_kernel = np.ones((3, 3), np.uint8)
    mask = cv2.erode(mask, small_kernel, iterations=1)
    kernel = np.ones((30, 30), np.uint8)
    mask = cv2.dilate(mask, kernel, iterations=3)

    #Convertir en una máscara sólida pq los grafitis se quedaban con huecos
    mask = solidify_mask(mask)

    cv2.imwrite(output_path, mask)
    crop_and_save_masks(output_path, pdf_name, "pink")
    return output_path
```

ANEXO V: GENERACIÓN MÁSCARA AMARILLA

```
def generate_yellow_mask(image_filename, pdf_name, lower_bound, upper_bound):
    input_path, output_path = get_paths(image_filename, pdf_name, "yellow")
    image = cv2.imread(input_path)
    if image is None:
        raise FileNotFoundError(f"No se pudo cargar la imagen en {input_path}")

    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv_image, lower_bound, upper_bound)

    # Kernel pequeño para la eliminar las líneas finas de azul que se cuelan
    small_kernel = np.ones((5, 5), np.uint8)
    mask = cv2.erode(mask, small_kernel, iterations=1)

    # Perform morphological operations to clean and fill the mask
    kernel = np.ones((30, 30), np.uint8)
    mask = cv2.dilate(mask, kernel, iterations=1)

    cv2.imwrite(output_path, mask)
    crop_and_save_masks(output_path, pdf_name, "yellow")
    return output_path
```