UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA CIVIL

TESIS

MACHINE LEARNING EN EL PROCESAMIENTO DIGITAL DE IMÁGENES RPA PARA LA AUTOMATIZACIÓN DE LA IDENTIFICACIÓN DE FALLAS EN EL PAVIMENTO

Para obtener el título profesional de Ingeniera Civil.

Elaborado por

BRIGITTE DIANA PUCHOC ESPEJO

ID: 0009-0006-0673-4904

Asesor

PhD. MIGUEL LUIS ESTRADA MENDOZA

ID: 0000-0002-8646-3852

LIMA - PERÚ

2025

Puchoc Espejo, Brigitte Diana

brigitte.puchoc.e@uni.pe

(+51) 932409418

DEDICATORIA

A mi papá Erasmo, mi mamá Zoraida
y a mis hermanos Karol, Frank y Luis. Los
amo mucho.

AGRADECIMIENTOS

Al Centro Peruano Japonés de Investigaciones Sísmicas y Mitigación de Desastres (CISMID), por haberme recibido por más de dos años en sus instalaciones y por brindarme sus equipos para el desarrollo de esta tesis. Me siento feliz de haber sido parte de tan prestigiosa institución.

Agradezco a mi asesor de tesis, el Dr. Miguel Luis Estrada Mendoza, quien, a pesar de tener una agenda muy ajustada, se dio el tiempo de revisar mis avances y corregirme de forma amable y constructiva, enseñándome siempre algo nuevo, tanto en lo profesional como en lo personal.

A esa persona cuya presencia fue como una luz en el camino, un motor en momentos decisivos. Gracias por aterrizar mis ideas, por hacer que creyera un poco más en mí, por ayudarme a darle forma a esta investigación, por acompañarme como co-asesoría silenciosa, por tu tiempo, y, sobre todo, por enseñarme que está bien aceptar ayuda a veces. Muchas gracias.

Agradezco también a todos mis amigos del CISMID que, aunque no lo sepan, han sido una parte importante para mí. Su presencia me ayudó a no sentirme sola, y aunque casi nunca hablábamos su compañía era suficiente. En especial, agradezco a mis, amigos del Laboratorio de Geomática: Fernando García, Oscar Solis, Carlos Dávila y Ángel Quesquén, por compartir todo este tiempo conmigo.

A toda mi familia por siempre esperar algo bueno de mí, quizá fue justamente esa expectativa la que me impulsó a seguir. A mi papá, por su trabajo arduo durante todos estos años, por todo ese esfuerzo y por ser el motor de mis sueños. A mi mamá, por impulsarme y empujarme cada vez que no creía en mí misma ni en que pudiera llegar tan lejos, y porque, a pesar de todo, siempre se sintió orgullosa de mí. A mi hermana Karol por quererme como si fuera una hija más para ella y por siempre corregirme y apoyarme en cada paso. A mi hermano Frank, que cuando yo no tenía el siempre me daba dinero a escondidas. A mi hermano Luis que, aunque no hablamos mucho, siempre supo darme sus consejos. A todos mis primos por hacerme reír cuando nos reunimos, en especial a mi primo Carlos, por su preocupación de siempre.

Finalmente, a todos mis compañeros de universidad, de todos los códigos, especialmente a Tito Vera, Maickie Mejía, Judith Mamani, Lesly Ortiz gracias por hacer más llevadera esta etapa, por sus consejos, por su tiempo, por aguantarme y por ser buenos amigos.

Resumen

La detección de fallas en pavimentos flexibles representa un desafío para el mantenimiento de la infraestructura vial. Las inspecciones tradicionales, basadas en observaciones visuales y técnicas manuales, suelen ser lentas, costosas y propensas a errores, dificultando una respuesta oportuna. Antes esta problemática, la presente investigación propone automatizar la detección de grietas, a través del uso de una red neuronal convolucional supervisada aplicada al análisis de ortomosaicos.

Para desarrollar este enfoque, se construyó una base de datos de 6,255 imágenes: 1,567 se generaron a partir de fuentes propias y 4,688 mediante técnicas de aumento de datos. Las imágenes fueron segmentadas manualmente, lo que permitió crear un conjunto de datos de entrenamiento para modelos de segmentación.

Con esta base de datos, se implementaron, entrenaron y compararon cinco arquitecturas de segmentación: U-Net, Shallow U-Net, VGG U-Net, ResNet101 U-Net y DeepLabV3+, evaluándolas mediante métricas de desempeño como IoU y F1-score, destacando ResNet101 U-Net por su mejor rendimiento, con un IoU de 0.693 y un F1-score de 0.816.

El modelo seleccionado fue aplicado sobre ortomosaicos georreferenciados de la Av. Argentina, comparando sus resultados con segmentaciones manuales en zonas representativas. Los resultados mostraron que la automatización redujo significativamente el tiempo de análisis, manteniendo un alto nivel de precisión en la segmentación de grietas en ortomosaicos, mostrando mayor eficiencia y optimizando el monitoreo vial de manera rápida y eficiente.

Palabras clave - Fallas en el pavimento, Ortomosaicos georreferenciados, Aprendizaje profundo, Sistemas de aeronaves pilotadas remotamente (RPAS), Visión por computadora

Abstract

The detection of distresses in flexible pavements poses a significant challenge for road infrastructure maintenance. Traditional inspections, based on visual observations and manual techniques, are often time-consuming, costly, and prone to human error, hindering timely decision-making. In response to this issue, the present research proposes the automation of crack detection using a supervised convolutional neural network applied to the analysis of georeferenced orthomosaics.

To support this approach, a dataset of 6,255 images was constructed: 1,567 images were derived from original sources, while 4,688 were generated through data augmentation techniques. All images were manually segmented, enabling the creation of a reliable training set for semantic segmentation models.

Using this dataset, five segmentation architectures were implemented, trained, and compared: U-Net, Shallow U-Net, VGG U-Net, ResNet101 U-Net, and DeepLabV3+. Their performance was evaluated using metrics such as Intersection over Union (IoU) and F1-score. ResNet101 U-Net achieved the best performance, with an IoU of 0.693 and an F1-score of 0.816.

The selected model was then applied to georeferenced orthomosaics of Av. Argentina, and its outputs were compared with manually segmented areas in representative zones. The results demonstrated that the automated method significantly reduced analysis time while maintaining high segmentation accuracy, thus improving efficiency and enabling rapid pavement condition monitoring.

Keywords - Pavement distresses, Georeferenced orthomosaics, Deep Learning, Remotely Piloted Aircraft Systems (RPAS), Computer vision

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Lista de Símbolos y Siglas

Símbolos

α : Tasa de aprendizaje inicial

σ : Varianza

λ : Factor de contraste

θ : Ángulo de rotación

β : Factor de brillo

x, y : Coordenadas en los ejes x y y

TP : Verdaderos positivos

T : Umbral

S : Tamaño del sensor

NB : Nivel de brillantez o intensidad de la imagen

N : Números de muestras

min_delta : Mínima diferencia para aplicar criterio de parada anticipada

L : Pérdida promedio

IT : Porcentaje de traslape por imagen.

I : Resolución de la imagen

H : Altura de vuelo sobre el terreno

FP : Falsos positivos

FN : Falsos negativos

F : Distancia focal de la cámara

D : Distancia recorrida del vehículo en 1 segundo.

C : Cantidad de imágenes que se generaron en 1 segundo.

A : Ancho captado por fotograma.

μ : Media

$\in$: Constante

θ_t : Parámetro actualizado en el paso t

β_2 : Factor de decaimiento del segundo momento

β_1 : Factor de decaimiento del primer momento

$z_{i,j,k}$: Valor de la neurona en la capa convolucional l

$x_{i',j',k'}$: Salida de la neurona en la capa l-1

$w_{u,v,k',k}$: Pesos del filtro convolucional

v_t : Momento de segundo orden en el paso t

val_loss_t : Pérdida de validación en el paso t

$val\_accuracy_t$: Precisión de validación en el paso t

s_h y s_w : Stride vertical y horizontal

m_t : Momento de primer orden en el paso t

g_t : Gradiente actual en el paso t

$f_n{}'$: Número de mapas de características en la capa anterior

f_h , f_w : Alto y ancho del campo receptivo

b_k : Término de sesgo (bias) para el mapa de características k

$\mathrm{y_i}$: Valor real de la muestra i

$\hat{y}_i$: Probabilidad predicha por el modelo para la clase positiva en la muestra i

$\hat{v}_t$: Momento de segundo orden corregido

$\hat{m}_t$: Momento de primer orden corregido

Siglas

ADAM : Adaptive Moment Estimation

ASPP : Atrous Spatial Pyramid Pooling

CISMID : Centro Peruano Japonés de Investigaciones Sísmica y Mitigación de Desastres

CMOS : Complementary Metal–Oxide–Semiconductor

CNN : Convolutional Neural Network

DEM : Digital Elevation Model

DL : Deep Learning

GCP : Ground Control Points

GIS : Geographic Information System

GNSS : Global Navigation Satellite System

GPU : Graphics Processing Unit

GSD : Ground Sample Distance

IA : Inteligencia Artificial

IoU : Intersection Over Union

ML : Machine Learning

MTC : Ministerio de Transportes y Comunicaciones

PCI : Pavement Condition Index

ReLU : Rectified Linear Activation Function

ResNet : Residual Network

RNN : Recurrent Neural Network

RPAS : Remotely Piloted Aircraft System

RTK : Real-Time Kinematic

SGD : Stochastic Gradient Descent

UAV : Unmanned Aerial Vehicle

VANT : Vehículo Aéreo No Tripulado

VGG : Visual Geometry Group

YOLO : You Only Look Once

Capítulo I. Introducción

1.1 Generalidades

El deterioro de los pavimentos flexibles es un problema recurrente en las infraestructuras viales, impactando directamente en la seguridad, la eficiencia del tránsito y los costos de mantenimiento. Entre las diversas manifestaciones de deterioro, las grietas representan una de las fallas más críticas, ya que permiten la infiltración de agua y agentes externos que aceleran el proceso de degradación estructural. La detección temprana y precisa de estas grietas es importante para implementar acciones de conservación que prolonguen la vida útil de las vías y optimicen la gestión de recursos destinados a su mantenimiento.

Tradicionalmente, la identificación de grietas en pavimentos se realiza mediante inspecciones visuales manuales, las cuales son susceptibles a errores humanos, demandan un esfuerzo considerable y presentan limitaciones en términos de cobertura y repetibilidad. En este contexto, el avance de las tecnologías de procesamiento de imágenes y el desarrollo de modelos de segmentación basados en aprendizaje profundo han abierto nuevas posibilidades para la automatización de estos procesos.

La presente investigación se enmarca en este contexto, enfocándose en la segmentación de grietas en pavimento flexible a partir de ortomosaicos georreferenciados obtenidos mediante imágenes capturadas por drones.

1.2 Descripción del problema de investigación

La seguridad vial es un tema crucial para reducir los accidentes de tráfico en nuestro país. Sin embargo, las soluciones solo se centran en reducir los niveles de velocidad dejando de lado las condiciones de la carretera, que también son una causa importante de los accidentes de tráfico (Dirección de Tecnologías de la Información y Comunicaciones, 2021).

Además de los riesgos para la seguridad, el mal estado del pavimento conlleva restricciones de capacidad en las carreteras y aumenta la congestión vehicular (Thomson & Bull, 2001). Los conductores tienden a reducir la velocidad o cambiar de carril para evitar las fallas, lo que ralentiza el flujo del tráfico. Por otro lado, las carreteras en buen estado permiten una circulación más fluida y rápida de los vehículos lo que reduce el tiempo de viaje y consumo de combustible.

La mala condición de la infraestructura vial no solo afecta la seguridad vial y la congestión, pues este es un factor importante en el crecimiento del país. La infraestructura vial es una precondición fundamental para el desarrollo social de cualquier país o estado, ya que el crecimiento económico de un país depende de los recursos disponibles para la sociedad y la eficiencia de su uso (Ivanova & Masarova 2013). La falta de mantenimiento de la infraestructura vial puede llevar a un deterioro acelerado del pavimento e incluso dañar las capas subyacentes, lo que aumenta el costo de reparación y disminuye la vida útil del pavimento. Realizar reparaciones oportunas puede evitar gastos mayores en la reconstrucción.

Además de los aspectos funcionales y económicos, el estado del pavimento tiene un impacto en la calidad del viaje y en la imagen de una ciudad. Un pavimento en mal estado resulta en una experiencia desagradable para peatones y conductores que transitan por largos tramos de carretera deteriorada. Por lo tanto, la reparación de fallas mejora la calidad del viaje y contribuye a una imagen positiva de la ciudad tanto para los residentes como para los visitantes.

El Ministerio de Transporte y Comunicaciones (MTC, 2022) informa que nuestro país cuenta actualmente con 30,209 km de carreteras pavimentadas. Estas carreteras tienen un tiempo de vida estimado de 10 años y casi la mitad de estas ya cumplieron dicha edad por lo que para alargar su tiempo de vida es necesario un mantenimiento preventivo o correctivo según sea el caso.

En la actualidad, contamos con una metodología establecida en el ASTM D6433 para determinar la condición del pavimento. Sin embargo, muchas de las metodologías convencionales en el mundo se consideran obsoletas, ya que implican inspecciones manuales que son ineficientes, consumen mucho tiempo y están limitadas por el sesgo humano y la experiencia. Además, su uso puede resultar costoso, lo que dificulta su aplicación.

El uso de estas metodologías convencionales implica una gran inversión de tiempo tanto para la recolección como para el procesamiento de datos. También implica la contratación de personal que se expone a riesgos de accidentes de tránsito al acercarse a tomar los datos. Además, cada evaluador tiene una perspectiva diferente al clasificar las fallas y asignarles una severidad, lo que puede afectar los resultados obtenidos.

Otro desafío importante es la gran congestión vehicular que existe en el país, especialmente en ciudades como Lima, que se encuentra entre las ciudades con mayor congestión vehicular en el mundo (TomTom, 2021), según el ranking Traffic Congestion Index del año 2021 de TomTom. Esto dificulta aún más la posibilidad de llevar a cabo el mantenimiento y evaluación de las carreteras, ya que cerrar las vías para la toma de datos sería una alternativa poco eficiente y generaría una mayor congestión vehicular.

El uso de metodologías convencionales para evaluar el estado del pavimento presenta varios desafíos en términos de eficiencia, seguridad, costos y congestión vehicular, por ello es necesario buscar enfoques más innovadores y tecnológicos que permitan una evaluación más precisa y eficiente de las condiciones de las carreteras, minimizando los riesgos para el personal y reduciendo la interrupción del tráfico. Esto facilitaría el mantenimiento oportuno de las carreteras y contribuiría a una infraestructura vial más segura y funcional en nuestro país ya que el monitoreo y mantenimiento de carreteras pavimentadas es una tarea costosa pero esencial para mantener una operación segura, por lo que considerando la demanda de una evaluación rápida del pavimento se deben establecer manera de garantizar el buen estado operativo de las carreteras y autopistas,

de manera eficiente y se regularice el monitoreo de manera económicamente viable por ello es necesario implementar una alternativa tanto para la toma de datos como para el procesamiento de éstos de manera rápida y eficiente, para que se pueda tomar decisiones en las reparaciones dando una prioridad de resane o un mantenimiento preventivo para evitar la extensión de las fallas. Además, los administradores de vías necesitan herramientas basadas en nuevas tecnologías que faciliten el trabajo y sobre todo disminuyan costos, tiempo y accidentes.

Actualmente, las imágenes satelitales pueden ser capaces de monitorear el peligro de la carretera. Sin embargo, debido al coste y a la calidad de resolución espacial limitada de la imagen, no se prefiere. Por lo tanto, las imágenes de diferentes plataformas, como los vehículos aéreos no tripulados (UAV), pueden ser rentables en comparación con las imágenes de satélite o la fotografía aérea tradicional.

Además el uso de drones se ha ido incrementando ya que permite una recolección de datos del terreno facilitando el trabajo, haciéndolos más rápidos y eficientes; asimismo el uso de la Inteligencia Artificial ha ido innovándose, automatizando el proceso de monitoreo de la superficie de la carretera, lo que puede resultar en grandes ahorros monetarios y puede conducir a ciclos de inspección más frecuentes, por esta razón los departamentos de mantenimiento, reparación y transporte de carreteras se han vuelto más interesados en usar sistemas automáticos para la evaluación del pavimento al punto de poder reconocer objetos, pero ¿Cómo puede el uso de aprendizaje profundo aplicado al procesamiento de imágenes aéreas capturadas por drones contribuir a la identificación confiable y automatizada de fallas en el pavimento flexible?

1.3 Objetivos del estudio

1.3.1 Objetivo general

Determinar la confiabilidad de emplear *Machine Learning* aplicado al procesamiento digital de imágenes de dron para identificar tipos de fallas en el pavimento flexible.

1.3.2 Objetivos específicos

- Recopilar y procesar imágenes de entrenamiento.
- Emplear algoritmos de procesamiento de imágenes para la automatización de la detección de fallas en el pavimento flexible.
- Comparar los resultados de *Machine Learning* con los resultados obtenidos de un levantamiento en campo.

1.4 Hipótesis

1.4.1 Hipótesis general

El uso de una arquitectura de red neuronal convolucional supervisada, como ResNet101 U-Net, aplicada al análisis de ortomosaicos generados a partir de imágenes aéreas capturadas por drones, permite identificar fallas en el pavimento flexible de manera confiable, alcanzando un rendimiento aceptable con una métrica IoU superior a 0.60.

1.4.2 Hipótesis específicas

- La recopilación y el procesamiento adecuado de imágenes aéreas capturadas mediante dron permiten generar un conjunto de datos de entrenamiento de calidad, suficiente para entrenar modelos de segmentación.
- El uso de Machine Learning logra identificar fallas en el pavimento flexible de manera automatizada.
- La comparación entre los resultados obtenidos mediante Machine Learning y los datos recolectados en campo permitirá evaluar de manera cualitativa la correspondencia entre ambos, proporcionando indicios sobre la confiabilidad del modelo en la detección de fallas en el pavimento.

1.5 Antecedentes investigativos

Silva et al. (2020) en su investigación realizaron un estudio en la provincia de Salamanca en España, buscando diseñar una plataforma que permita la detección de daños en rutas

de transporte mediante un sistema multiagente y drones equipados con algoritmos de detección. Para ello iniciaron capturando 568 imágenes con el dron DJI Air 2 a 60m del suelo, realizando posteriormente un etiquetado diferenciando baches, grietas y delimitando cada clase, luego ampliaron su data modificando la orientación y el tamaño de la imagen, obteniendo un total 1362 imágenes, de las cuales usaron el 70% para el entrenamiento del programa, el 20% para la validación y el 10% para probar la efectividad del modelo entrenado. A continuación, entrenaron al modelo en una red DarkNet utilizando un conjunto de datos diseñado específicamente para carreteras europeas. Los resultados que obtuvieron fueron favorables pues obtuvieron un 95% de precisión en la detección de anomalías.

Zúñiga Guisado (2022) desarrolló un estudio en el distrito de Villa María del Triunfo, Lima, con el objetivo de detectar fallas en pavimentos como grietas y huecos mediante el uso de *Deep Learning*. La metodología consistió en la recopilación de 420 imágenes, divididas en 292 imágenes de grietas y 128 imágenes de huecos, capturadas con un teléfono inteligente en las principales calles del área de José Carlos Mariátegui. Se utilizó el modelo YOLOv5 y el lenguaje de programación Python, junto con herramientas como OpenCV, PyTorch y TensorBoard. El estudio reportó resultados con una precisión de 0.93 para grietas y 0.77 para huecos, mientras que la sensibilidad fue de 0.92 y 0.91 respectivamente. La investigación concluyó que el uso de redes neuronales convolucionales permite automatizar la detección de fallas viales de manera eficiente, facilitando la planificación del mantenimiento y la toma de decisiones.

Ignacio (2022) realizó una investigación en Pimentel, Perú, con el objetivo de identificar automáticamente grietas en pistas de asfalto mediante procesamiento digital de imágenes. El estudio utilizó una población de 500 imágenes, seleccionando una muestra del 80% (400 imágenes) para el análisis. Las imágenes fueron capturadas con una cámara digital con resolución de 5888x3584 píxeles y divididas en fragmentos de 256x256 píxeles. La metodología incluyó preprocesamiento de imágenes, donde se aplicaron técnicas como el

aumento del brillo y la umbralización binaria para mejorar la visualización de las grietas. Posteriormente, se implementó un algoritmo de Red Neuronal Convolucional (CNN), logrando una exactitud del 98%, una sensibilidad del 88% y una especificidad del 96%. El tiempo de procesamiento promedio fue de 85 segundos por imagen. Los resultados demostraron que la propuesta es eficiente para la detección de grietas, facilitando el mantenimiento preventivo de la infraestructura vial.

Román-Garay et al. (2023) desarrollaron un estudio en la ciudad de Culiacán, México, con el objetivo de implementar una metodología para la detección y segmentación semántica de grietas y baches en pavimentos flexibles mediante redes Transformer. La investigación utilizó una base de datos compuesta por 245 imágenes, las cuales fueron ampliadas mediante técnicas de aumento de datos como rotación, obteniendo un total de 2,052 imágenes. Las imágenes fueron capturadas con una cámara GoPro Hero 8 Black, con resolución de 4000x3000 píxeles, montada a 1.20 metros de altura para cubrir el ancho del pavimento en recorridos sistemáticos por calles específicas de la ciudad. La metodología incluyó un preprocesamiento avanzado aplicando cambios en el contraste y conversión a escala de grises para mejorar la calidad de las imágenes y reducir el impacto de ruido como sombras e iluminación inconsistente. Posteriormente, se implementó la arquitectura SegFormer, una red basada en Transformers, para realizar la segmentación semántica de grietas y baches. El modelo alcanzó métricas destacadas, con una precisión del 82.35%, un Recall del 96.55% y un F1-Score de 88.89%, demostrando robustez en ambientes no controlados y bajo diferentes condiciones ambientales.

Chino e Inchicsana (2024) realizaron una investigación en el distrito de Socabaya, Arequipa, Perú, con el objetivo de desarrollar una aplicación móvil que utiliza algoritmos de *Deep Learning* para la detección de anomalías en pavimentos, tales como grietas, desprendimiento de agregados y piel de cocodrilo. La metodología incluyó la recopilación de 250 imágenes tomadas con teléfonos inteligentes en calles específicas pavimentadas con asfalto. Se utilizó la arquitectura YOLOv5 entrenada con un conjunto de datos dividido

en 200 imágenes para entrenamiento, 100 para validación y 30 para prueba. El modelo alcanzó precisiones del 93% para grietas y del 77% para huecos, con un F1-score de 0.92 para grietas y 0.91 para huecos. Como resultado, se concluyó que la implementación del sistema basado en redes neuronales convolucionales es efectiva y permite una detección automatizada de fallas, mejorando la planificación y el mantenimiento de la infraestructura vial.

Kondo y Ukita (2021) desarrollaron un método para la segmentación de grietas en imágenes de baja resolución utilizando una técnica de aprendizaje conjunto con Super-Resolución (SR) y Segmentación Semántica (SS). El objetivo principal fue obtener resultados de segmentación con la misma calidad que aquellos logrados en imágenes de alta resolución, resolviendo problemas como la baja calidad visual y el desequilibrio de clases en las grietas. La metodología combinó dos redes principales: una red de super-resolución basada en Deep Back-Projection Network y una red de segmentación semántica basada en U-Net. La arquitectura fue entrenada y validada en el conjunto de datos Khanhha, compuesto por 9,603 imágenes para entrenamiento y 1,695 imágenes para prueba, que incluyen imágenes de alta resolución con grietas de diferentes espesores. Los resultados experimentales demostraron que el método propuesto logró un valor de *Intersection Over Unión* (IoU) máxima de 57.1% en imágenes de baja resolución, alcanzando una precisión comparable a la segmentación realizada en imágenes de alta resolución.

Silva (2019) realizó un estudio enfocado en la detección automática de grietas en puentes de hormigón a través del uso de modelos *Encoder-Decoder* de segmentación semántica. El objetivo principal fue implementar y optimizar modelos de *Deep Learning* para localizar y segmentar grietas a partir de imágenes capturadas de puentes de concreto, con énfasis en el uso de arquitecturas SegNet, *Encoder-Decoder* with *Skip Connections* y DeepLabV3+ .El conjunto de datos utilizado estuvo compuesto por imágenes recopiladas mediante vehículos aéreos no tripulados (UAVs) y procesamiento manual para la generación de

etiquetas. Las imágenes, obtenidas desde distintos ángulos y resoluciones, fueron utilizadas para entrenar y validar los modelos implementados en TensorFlow. Se realizaron múltiples experimentos con ajustes en hiperparámetros y front-ends preentrenados como ResNet50/101/152 y MobileNetV2, evaluando métricas como IoU, precisión y F1-Score. Los resultados demostraron que la arquitectura DeepLabV3+ superó en rendimiento a las otras configuraciones evaluadas, logrando una IoU promedio del 84% y una precisión del 87% en la segmentación de grietas. En contraste, las arquitecturas SegNet y *Encoder-Decoder* with *Skip Connections* alcanzaron una IoU del 74% y 79%, respectivamente. El estudio destacó la efectividad de los modelos *Encoder-Decoder* para segmentar grietas con alta precisión, especialmente en escenarios complejos donde la iluminación y el ruido afectan la calidad de las imágenes.

Orellana (2019) desarrolló un estudio en Chile con el objetivo de comparar y evaluar la eficacia de modelos de clasificación y segmentación semántica aplicados a la detección automática de grietas en concreto, especialmente en puentes. La investigación utilizó las arquitecturas DeepLabV3+ y *Encoder-Decoder* con y sin conexiones tipo *Skip*, aplicadas a un conjunto de datos compuesto por 1,638 imágenes capturadas de archivos de video. Durante el preprocesamiento, las imágenes fueron segmentadas manualmente para generar las etiquetas de entrenamiento. Posteriormente, se realizaron experimentos variando los parámetros de entrenamiento, como el tamaño del lote, la tasa de aprendizaje y el uso de pesos preentrenados en ResNet50 como *encoder*. Los resultados demostraron que la arquitectura DeepLabV3+ alcanzó un IoU promedio del 89%, mostrando un buen rendimiento general. Por otro lado, la arquitectura *Encoder-Decoder* con *Skip Connections* logró un IoU ligeramente superior, con un 90%, aunque presentó un mayor tiempo de procesamiento en comparación con DeepLabV3+. La investigación concluyó que ambas arquitecturas son eficaces para la detección de grietas, destacando que la selección del modelo dependerá del balance entre precisión y eficiencia temporal.

Kulkarni et al. (2022) desarrollaron un estudio enfocado en la segmentación de grietas mediante la creación de un conjunto de datos denominado CrackSeg9k, compuesto por 9,255 imágenes de diferentes superficies, como pavimentos, muros de mampostería, cerámica y concreto. El conjunto de datos fue generado a partir de la combinación de varios conjuntos de datos públicos existentes, realizando un proceso de refinamiento en las etiquetas para garantizar la consistencia y precisión en las anotaciones. La metodología del estudio incluyó un preprocesamiento de imágenes, ajustando la resolución a 400x400 píxeles y aplicando operaciones morfológicas, como la erosión y dilatación, con el objetivo de eliminar ruido y corregir bordes distorsionados en las anotaciones. Las imágenes fueron segmentadas en función de la complejidad de las grietas, clasificándolas en tres categorías: lineales, ramificadas y enmarañadas. Posteriormente, se implementaron modelos de *Deep Learning* para la segmentación, evaluando arquitecturas como DeepLabV3+ con ResNet101, Pix2Pix y Swin U-Net. Los resultados del estudio demostraron que DeepLabV3+ con ResNet101 alcanzó los mejores valores de rendimiento, con un IoU de 0.7712 y un F1-Score de 0.7238, superando a modelos como Pix2Pix y Mask-RCNN.

En esta propuesta se busca introducir mejoras importantes respecto a estudios previos que analizaron fallas en pavimentos, tanto en la cantidad de datos como en la metodología utilizada y el nivel de automatización.

En primer lugar, la base de datos generada en esta investigación supera a la de los antecedentes revisados, al contar con un total de 1567 imágenes, una cantidad mayor al generado en otros estudios. Este aspecto es fundamental, ya que el entrenamiento con una base de datos propia permite una mejor adaptación a las fallas presentes en el pavimento de nuestro país, optimizando así su capacidad de detección y precisión. Además, es importante mencionar que, aunque las investigaciones previas presentaron cantidades significativas de imágenes, muchas de estas buscaron detectar múltiples tipos de fallas, como grietas, huecos, desprendimiento de agregados y piel de cocodrilo. Esta

decisión redujo la cantidad de datos destinados a cada tipo de falla, lo que podría afectar la capacidad de generalización del modelo en la detección de una falla específica. En contraste, la presente investigación se enfoca exclusivamente en la segmentación de fallas, en general permitiendo así una detección más precisa y robusta.

En cuanto a la adquisición y procesamiento de imágenes, la presente investigación plantea una metodología más variada que la utilizada en estudios anteriores, realizando recortes al ortomosaico obtenido mediante imágenes capturadas con dron, seguido de un proceso de segmentación manual de cada recorte. Adicionalmente, con el fin de ampliar la base de datos, se utilizaron imágenes obtenidas con cámaras, extraídas de frames de videos, lo que permitió incrementar la cantidad de datos con información adicional relevante, garantizando una combinación entre calidad y representatividad de los datos empleados en el modelo.

En cuanto a la metodología aplicada, la presente investigación introduce un enfoque innovador y preciso mediante el uso de segmentación semántica para la detección de fallas en pavimento flexible. Este proceso automatizado mejora significativamente la precisión y continuidad en la detección, superando los enfoques tradicionales basados en detección de objetos empleados en algunas de las investigaciones anteriores.

En conclusión, esta investigación no solo aborda las limitaciones identificadas en trabajos anteriores, sino que también introduce mejoras significativas en términos de cantidad y calidad de datos, metodología de segmentación y automatización del proceso, posicionándose como una mejor propuesta para la segmentación de fallas en pavimento flexible.

Capítulo II. Marco teórico y conceptual

2.1 Pavimento

El *Manual de Carreteras: Suelos, Geología, Geotecnia y Pavimentos*, elaborado por el Ministerio de Transportes y Comunicaciones (2014), define al pavimento como una estructura de varias capas que se usa para resistir y distribuir los esfuerzos originados por los vehículos, así también brinda comodidad y seguridad para los usuarios de la vía.

Según Coronado (2002), este viene a ser una estructura cuya función es sostener las cargas vehiculares, la cual está conformada por la subrasante, subbase, base y carpeta.

Montejo (2006) define al pavimento como un conjunto de capas superpuestas de manera horizontal, diseñadas y construidas con materiales apropiados que se encuentra compactados entre sí, todo ello para poder resistir los esfuerzos de las cargas repetidas originadas por el tránsito.

De todo lo anterior concluimos que el pavimento es una estructura que se utiliza para soportar y distribuir las cargas de tráfico de manera eficiente, además poseen varias capas que le permiten mejorar su rendimiento ante estas cargas, todo ello con la finalidad de facilitar el tránsito, hacerlo más cómodo, seguro y eficiente.

2.1.1 Clasificación de los pavimentos

El *Manual de Carreteras: Suelos, Geología, Geotecnia y Pavimentos*, elaborado por el Ministerio de Transportes y Comunicaciones (2014), tenemos tres tipos de pavimentos:

2.1.1.1 Pavimento flexible.

El pavimento flexible es una estructura compuesta por una capa de rodadura o carpeta asfáltica compuesta de materiales bituminosos y capas granulares como la base y la subbase, véase Figura 1.

La carpeta asfáltica cumple tres funciones principales, la primera es brindar comodidad a través de su superficie de rodadura estable al tránsito, la segunda es poseer una resistencia adecuada que complemente su capacidad estructural y por último su permeabilidad, impidiendo en lo posible el paso del agua al interior del pavimento.

La base granular transmite los esfuerzos producidos por el tránsito a la subbase y a la subrasante en una intensidad apropiada.

La subbase granular actúa como capa de transición entre la base y la subrasante impidiendo el paso de las partículas finas disminuyan la calidad de la base, así también se adapta a los cambios que sufre la subrasante, impidiendo que estas deformaciones se transmitan a la superficie de rodamiento, por último, transmite los esfuerzos en menor intensidad que la base, pero haciendo la que subrasante reciba cargas en un nivel adecuado.

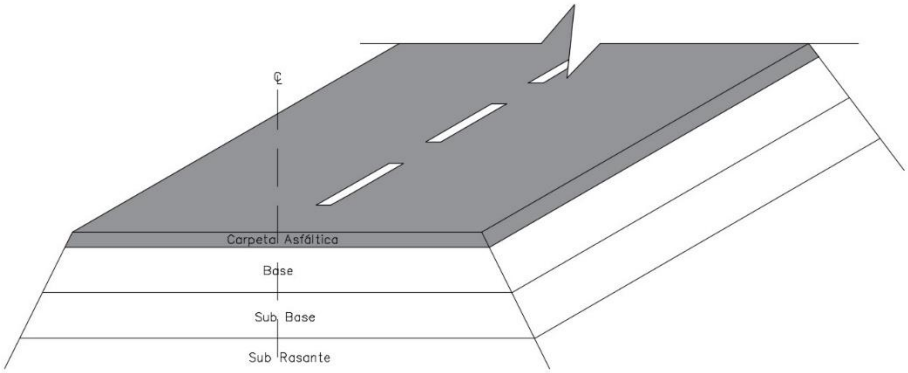


Figura 1: *Estructura típica de un pavimento flexible*. Adaptado de Montejo (2006).

2.1.1.2 Pavimento semirrígido.

Según el Manual de Carreteras, Suelos, Geología, Geotecnia y Pavimentos (2014), es una estructura compuesta por capas asfálticas; estas están compuestas por una carpeta asfáltica sobre una base tratada con asfalto, cemento o cal y dentro de esta definición se incluyen a los pavimentos adoquinados.

2.1.1.3 Pavimento rígido

Según el Manual de Carreteras, Suelos, Geología, Geotecnia y Pavimentos (2014), el pavimento rígido es una estructura compuesta por una capa de subbase granular o base granular y una capa de rodadura de losa de concreto de cemento hidráulico como aglomerante, véase Figura 2.

La losa de concreto además de cumplir las mismas funciones de la carpeta asfáltica, también soportar cargas y las transmite en niveles adecuados.

La subbase impide el paso del suelo fino a la superficie bloqueando el ingreso de agua por las juntas de las losas, también sirve como capa de transición, suministra apoyo al pavimento, mejora el drenaje y ayuda a controlar los cambios de volumen de la subrasante para que dichos cambios no afecten a la losa.

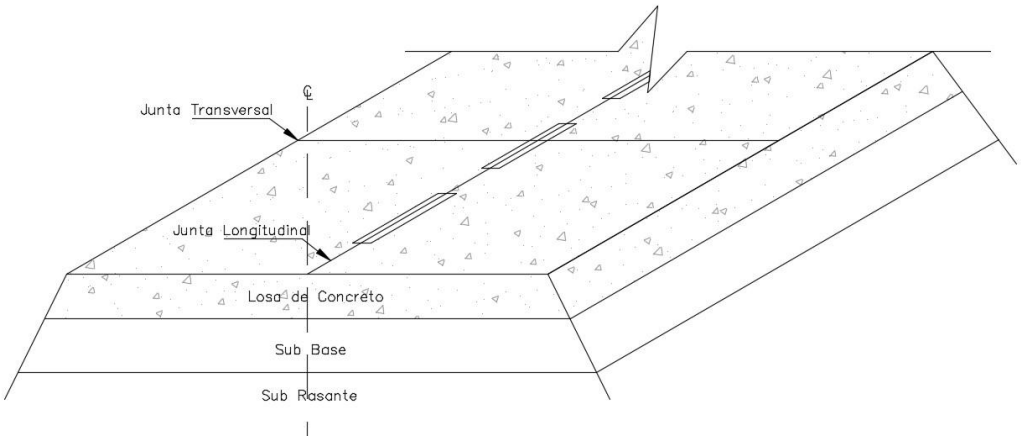


Figura 2: Estructura Típica de un Pavimento Rígido. Adaptado de Montejo (2006).

2.1.2 Definición y tipos de grietas en pavimentos asfálticos

El manual del PCI (*Pavement Condition Index*), escrito por Vásquez (2002) y basado en la norma ASTM 6433 "Standard Practice for Roads and Parking Lots Pavement Condition Index Surveys", define las grietas en pavimentos flexibles como discontinuidades o fisuras en la superficie asfáltica que pueden comprometer su integridad estructural. Estas grietas pueden ser causadas por diversos factores, como la fatiga del material debido a cargas repetitivas, contracción térmica, movimientos de la base o deficiencias en la construcción.

Este manual considera en total 19 fallas en el pavimento flexible. De estas, seis se clasifican específicamente como grietas:

- Piel de cocodrilo: Red de grietas interconectadas con forma de malla, originadas por fatiga del pavimento debido a cargas repetidas del tránsito.
- Agrietamiento en bloque: Grietas que dividen el pavimento en bloques de tamaño regular, causadas principalmente por contracción térmica y envejecimiento del asfalto.
- Grieta de borde: Fisura que se desarrolla paralelamente al borde del pavimento, generalmente debido a la debilidad de la base o la subrasante.
- Grieta de reflexión de junta: Se produce sobre juntas preexistentes en capas subyacentes, como losas de concreto, debido a movimientos térmicos.
- Grietas parabólicas: Grietas en forma de medialuna causadas por esfuerzos de frenado o aceleración en una mezcla asfáltica de baja resistencia.
- Grietas longitudinales y transversales: Se extienden en paralelo o perpendicularmente al eje del pavimento y pueden ser causadas por contracción del material o problemas en la construcción de juntas.

2.2 Fotogrametría aérea

Según el Instituto Geográfico Nacional de España (IGN, 2016), la fotogrametría aérea se define como una técnica que permite obtener información precisa sobre la forma, dimensiones y posición de objetos en la superficie terrestre mediante el análisis de fotografías tomadas desde plataformas aéreas, como aviones o drones. Esta técnica se basa en la captura de imágenes con cierto solapamiento, lo que facilita la reconstrucción tridimensional del terreno y la generación de productos cartográficos precisos.

Uno de los principales productos derivados de la fotogrametría aérea es el ortomosaico, una imagen compuesta que resulta de la unión de múltiples fotografías aéreas corregidas

geométricamente para eliminar distorsiones causadas por la perspectiva. Este proceso de corrección es conocido como ortorrectificación, el cual asegura que cada punto del ortomosaico corresponda a coordenadas geográficas precisas.

2.2.1 Tecnología de drones en fotogrametría

Los Drones son también conocidos como Vehículos Aéreos no Tripulados (UAV, por sus siglas en inglés), o RPAS (Remote Pilot Aerial System), son aeronaves que operan a control a distancia, es decir sin la necesidad de un piloto a bordo (Jiménez, 2020). Originalmente desarrollados para aplicaciones militares, su uso se ha expandido a diversas áreas, incluyendo la fotogrametría aérea, donde se emplean para capturar imágenes detalladas del terreno desde una perspectiva aérea.

En fotogrametría, los drones se clasifican en multirrotores los cuales se encuentran equipados con múltiples hélices, y drones de ala fija, que poseen una estructura similar a la de los aviones tradicionales. La elección entre ambos depende de las necesidades del proyecto, considerando factores como el área a cubrir, la autonomía de vuelo y las condiciones del terreno.

Para la captura de imágenes, los drones pueden incorporar sensores ópticos, como cámaras RGB, utilizadas en la generación de ortomosaicos, y sensores LiDAR, empleados en la reconstrucción tridimensional del terreno. La selección del sensor está determinada por los requerimientos del estudio y la precisión esperada en el procesamiento de datos.

2.2.2 Georreferenciación y ortorrectificación de imágenes aéreas

La georreferenciación y la ortorrectificación son procesos en la fotogrametría aérea, que garantizan la precisión espacial y geométrica de las imágenes obtenidas mediante drones.

2.2.2.1 Georreferenciación de imágenes aéreas

La georreferenciación consiste en asignar coordenadas espaciales a una imagen ráster, permitiendo que cada píxel corresponda con una ubicación específica en la superficie

terrestre. Para lograrlo, tradicionalmente se emplean puntos de control terrestre (GCP por sus siglas en inglés, Ground Control Points), que sirven como referencias en el sistema de coordenadas deseado. Sin embargo, con la evolución de la tecnología de posicionamiento, los drones modernos han incorporado un sistema global de navegación por satélite (GNSS por sus siglas en inglés, *Global Navigation Satellite System*) avanzados con módulos RTK (*Real-Time Kinematic*), los cuales permiten obtener imágenes georreferenciadas con alta precisión sin necesidad de GCP (Pazmiño, 2019). Este sistema GNSS utiliza señales de un junto con una estación base o red de referencia para corregir errores de posicionamiento, eliminando gran parte de las distorsiones que afectan las imágenes aéreas convencionales.

2.2.2.2 Ortorrectificación de imágenes aéreas

La ortorrectificación es el proceso mediante el cual se corrigen las distorsiones geométricas presentes en las imágenes aéreas, ocasionadas por la inclinación de la cámara, la curvatura terrestre y la topografía del terreno. Mediante este procedimiento, se elimina la deformación de perspectiva, generando imágenes donde todas las escalas son uniformes y las mediciones de distancias y áreas son precisas (Urrego & Castro, 2018).

Para la orto rectificación, es necesario un Modelo Digital de Elevación (DEM por sus siglas en inglés, Digital Elevation Model), el cual representa la topografía del terreno y permite ajustar la imagen conforme a su relieve. Como resultado, se obtiene un ortomosaico, que es una imagen compuesta por múltiples fotografías corregidas geométricamente, asegurando una representación precisa del área de estudio.

2.2.3 Ortomosaicos y su relación con el GSD

Un ortomosaico es una imagen compuesta generada a partir de múltiples fotografías aéreas, las cuales han sido ortorrectificadas para corregir distorsiones geométricas y alineadas en un mosaico continuo. Además, se le aplica un ajuste de color para garantizar la uniformidad visual en toda la imagen (Esri, 2023).

El proceso de generación de un ortomosaico se basa en la captura de imágenes aéreas con cierto grado de solapamiento, lo que permite reconstruir la superficie terrestre en función de la información redundante entre las fotografías. Posteriormente, las imágenes se someten a un proceso de ortorrectificación, el cual corrige las distorsiones causadas por la inclinación de la cámara y la variabilidad en la topografía del terreno.

Para garantizar la calidad del ortomosaico, se recomienda un solapamiento de 80 % longitudinal y entre 60-80 % lateral, valores que aseguran una cobertura óptima y minimizan la aparición de errores en el ensamblado de las imágenes. Un solapamiento insuficiente puede generar inconsistencias en la alineación de las fotografías, mientras que un solapamiento excesivo puede aumentar innecesariamente el volumen de datos y el tiempo de procesamiento. (Escalante et al., 2016)

Un parámetro importante en la calidad del ortomosaico es la Distancia de Muestra en el Terreno (GSD por sus siglas en inglés, *Ground Sample Distance*), la cual define la representación espacial de cada píxel en la imagen final. El GSD se expresa en unidades de longitud (por ejemplo, centímetros por píxel) e indica la distancia real en el suelo que cubre cada píxel de la imagen. Un GSD menor implica una mayor precisión en el estudio, ya que la resolución de la imagen está directamente relacionada con el nivel de detalle capturado. La exactitud de los resultados no puede superar el valor del GSD establecido. (Carvajal et al., 2021)

Su valor está determinado por la resolución del sensor de la cámara, la altura de vuelo del dron y la distancia focal del lente. La ecuación general para calcular el GSD es la siguiente:

$$GSD = \frac{H \times S}{F \times I}$$

Donde:

GSD: es la distancia de muestra en el terreno (cm/píxel ó m/pixel)

H: altura de vuelo sobre el terreno (m ó cm)

S: tamaño del sensor (mm)

F: distancia focal de la cámara (mm)

I: resolución de la imagen (píxeles)

2.2.4 Ventajas y desafíos de la fotogrametría aérea con dron

La fotogrametría aérea con drones ha revolucionado la captura de datos geoespaciales, ofreciendo múltiples ventajas frente a los métodos tradicionales. Según González et al. (2019) una de sus principales ventajas es la seguridad, ya que permite recolectar información sin necesidad de trasladarse a zonas peligrosas, reduciendo significativamente los riesgos en terrenos inestables o de difícil acceso.

Así también se destaca la capacidad de generar bases de datos más amplias, dado que las imágenes obtenidas no solo permiten analizar la zona de interés, sino también su entorno inmediato. Esto contribuye a una representación más realista del terreno, proporcionando datos más completos y precisos en comparación con los métodos convencionales.

Además, la fotogrametría aérea con drones permite dar continuidad al monitoreo de la zona estudiada, posibilitando la generación de series temporales que facilitan el análisis de cambios en el tiempo. Gracias a la rapidez y facilidad en la recolección de datos, se pueden realizar levantamientos periódicos con menor esfuerzo.

Desde una perspectiva económica, esta tecnología es más accesible y menos costosa que otras soluciones tradicionales. La implementación de drones reduce los costos asociados al reconocimiento y levantamiento de información de terrenos, obras y áreas extensas, optimizando el tiempo y los recursos utilizados.

No obstante, a pesar de sus múltiples ventajas, la fotogrametría aérea con drones presenta ciertas limitaciones que deben considerarse en la planificación de un levantamiento.

Uno de los principales desafíos es la dependencia de las condiciones meteorológicas. Factores como la lluvia, la niebla y vientos superiores a 30 km/h pueden afectar el control del dron, comprometiendo la estabilidad de la aeronave y la calidad de las imágenes capturadas. Estos elementos pueden generar distorsiones en la captura o incluso impedir la realización del vuelo, lo que limita su operatividad en entornos climáticos adversos (Fernández & Gutiérrez, 2016).

Asimismo, la autonomía del dron es un factor determinante, ya que la duración de la batería restringe el tiempo de vuelo continuo, lo que puede requerir múltiples vuelos o pausas para recargar o reemplazar baterías.

Otro aspecto a considerar es la influencia de la iluminación natural en la calidad de las imágenes georreferenciadas. Las variaciones en la intensidad de la luz solar a lo largo del día y las diferencias estacionales entre luces y sombras pueden afectar la visibilidad de ciertos elementos del terreno. Esto puede generar inconsistencias en la apariencia de las imágenes y afectar la precisión de los modelos generados a partir de ellas.

2.3 Inteligencia artificial

Según García (2012), la Inteligencia Artificial (IA) es “un conjunto de técnicas, algoritmos y herramientas que nos permiten resolver problemas para los que, a priori, es necesario cierto grado de inteligencia, en el sentido de que son problemas que suponen un desafío incluso para el cerebro humano” (p. 7). Por otro lado Rouhiainen (2018) describen la IA como la capacidad de las máquinas para utilizar algoritmos, aprender de los datos y aplicar lo aprendido en la toma de decisiones de manera autónoma, tal y como lo haría un ser humano, pero sin necesidad de descanso, y con la capacidad de analizar grandes volúmenes de información simultáneamente, con una menor tasa de error.

De lo anterior, se puede decir que la IA como una tecnología diseñada para replicar capacidades humanas, con la ventaja de procesar grandes cantidades de datos y mejorar su rendimiento a lo largo del tiempo.

Dentro de la inteligencia artificial encontramos a el Aprendizaje Automático o también conocido como aprendizaje máquina (ML por sus siglas en inglés, Machine Learning), que busca identificar y generalizar patrones a partir de muestras, mejorando su desempeño a medida que se entrene con una mayor cantidad de datos (Bishop, 2006). Este aprendizaje lo realiza mediante el uso de algoritmos avanzados que permiten procesar grandes volúmenes de datos históricos, lo que le permite ser capaz de realizar predicciones precisas.

Una subdisciplina de *Machine Learning* es el Aprendizaje Profundo o también llamado *Deep Learning* (DL) que se centra en el desarrollo de modelos capaces de generar nuevos datos que sean indistingible por humanos a través del uso de técnicas avanzadas de aprendizaje profundo. (Velasco, 2024)

2.3.1 Principales enfoques en *Deep Learning*

En el ámbito del aprendizaje profundo, se han desarrollado diversas arquitecturas de redes neuronales para abordar distintos tipos de datos y tareas. Entre las más destacadas se encuentran las Redes Neuronales Recurrentes (RNN), los Transformers y las Redes Neuronales Convolucionales (CNN).

Las RNN, están diseñadas para procesar datos secuenciales, como series temporales o secuencias de texto. Su arquitectura incluye conexiones recurrentes que les permiten mantener una "memoria" de estados anteriores, lo que es esencial para tareas como traducción automática y reconocimiento de voz (Goodfellow, Bengio & Courville, 2016).

Las transformes por su parte han revolucionado el procesamiento de secuencias al eliminar la necesidad de procesamiento secuencial inherente en las RNN. Utilizan mecanismos de autoatención que permiten modelar relaciones a largo plazo de manera más eficiente, facilitando el procesamiento en paralelo y mejorando el rendimiento en tareas de procesamiento de lenguaje natural (Vaswani et al., 2017).

Las CNN han demostrado ser especialmente efectivas en el procesamiento de datos con estructuras de tipo rejilla, como imágenes y videos. Su arquitectura se basa en la aplicación de convoluciones, que son operaciones matemáticas que permiten extraer características locales de los datos. Una CNN típica consta de varias capas, incluyendo capas convolucionales, que aplican filtros para detectar características locales, como bordes o texturas, capas de agrupamiento (*pooling*), que reducen la dimensionalidad de las características extraídas, manteniendo la información más relevante y disminuyendo la carga computacional y capas completamente conectadas que integran las características extraídas para realizar la clasificación o regresión final.

2.3.2 Técnicas de *Deep Learning*

El aprendizaje profundo ha desarrollado diversas técnicas para abordar problemas complejos en diferentes dominios, especialmente en visión por computadora. Entre las técnicas más utilizadas se encuentran la clasificación de imágenes, la detección de objetos y la segmentación de imágenes, cada una con aplicaciones específicas.

La clasificación de imágenes es una de las técnicas que permite asignar una etiqueta a una imagen completa y facilitando la identificación de su contenido principal. Modelos basados en CNN han logrado superar el rendimiento humano en tareas de reconocimiento visual, alcanzando precisiones sorprendentes en bases de datos como ImageNet (LeCun et al., 2015). A pesar de su éxito, la clasificación no permite determinar la ubicación exacta de los objetos dentro de una imagen, lo que llevó al desarrollo de técnicas más avanzadas como la detección de objetos.

La detección de objetos no solo identifica la presencia de un objeto en una imagen, sino que también localiza su posición mediante cuadros delimitadores. Esta técnica ha sido ampliamente adoptada en aplicaciones que requieren un análisis detallado del entorno, como los sistemas de videovigilancia y los vehículos autónomos (Redmon et al., 2016).

Mientras que la detección de objetos proporciona información sobre la ubicación general de un objeto en una imagen, la segmentación de imágenes permite una identificación más precisa al clasificar cada píxel de la imagen, generando una representación detallada de los objetos en la escena. Dependiendo del enfoque utilizado, la segmentación puede ser semántica, de instancias o panóptica. La segmentación semántica asigna una etiqueta a cada píxel sin distinguir entre diferentes instancias del mismo objeto, lo que resulta útil en aplicaciones como la segmentación de carreteras en mapas de navegación. En contraste, la segmentación de instancias distingue entre diferentes ejemplares de una misma clase, permitiendo, por ejemplo, la identificación individual de automóviles en una imagen de tráfico (Kirillov et al., 2019). La segmentación panóptica combina ambas aproximaciones, proporcionando una comprensión completa de la escena al clasificar todos los píxeles y diferenciar entre distintos objetos de la misma categoría, estos tipos de segmentaciones podemos observarlas en la Figura 3, donde se encuentran ejemplos de clasificación con localización (b), detección de objetos (c) y segmentación semántica (d), todo ello aplicado a una imagen del pavimento (a).

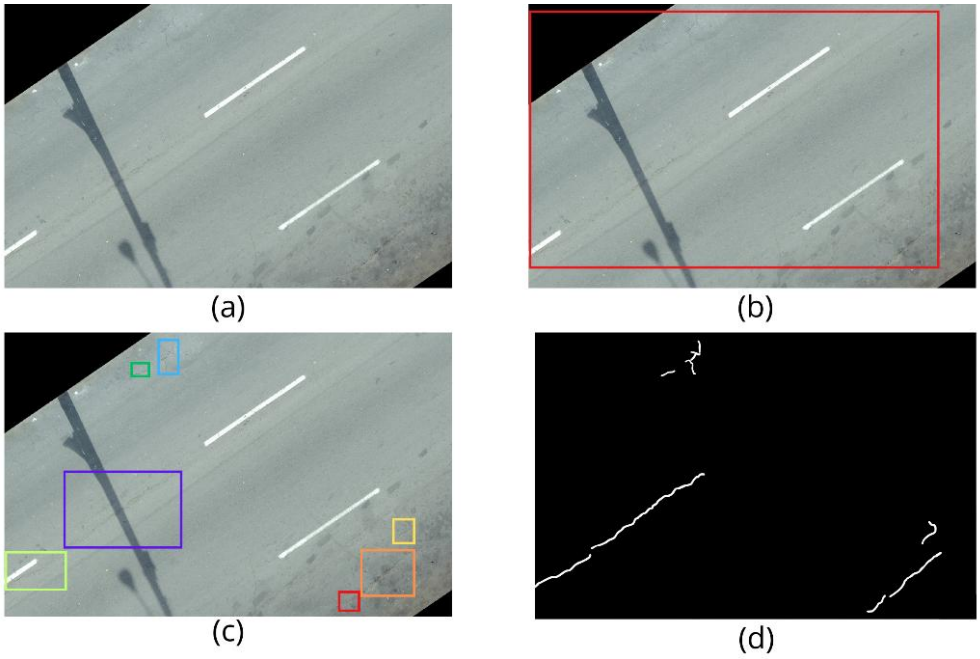


Figura 3: Técnicas de Deep Learning: clasificación con localización, detección de objetos y segmentación semántica.

2.3.3 Visión por computadora

La visión por computadora es una disciplina que permite a las máquinas interpretar y procesar imágenes del mundo real, imitando la capacidad de visión humana. Una imagen bidimensional puede definirse como una función que asigna a cada par de coordenadas (x, y) un valor correspondiente a una característica específica del punto representado, el cual se le conoce como píxel (del inglés, *picture element*)., como su nivel de brillo o tonalidad (Jiménez, 2000), como se observa en la siguiente ecuación:

$$NB = f(x, y)$$

Donde:

NB: Nivel de brillantez o intensidad de la imagen

x,y: Coordenadas o ubicación del píxel dentro de la imagen

Existen diferentes formas de organizar o codificar los diferentes colores a partir de componentes básicas, lo que se conoce como espacios de color. Por ejemplo, una imagen en RGB se basa en los tres sensores humanos, considerando que todos los colores son una combinación de tres colores básicos: R (rojo), G (verde), B (azul).

2.3.3.1 Técnicas de procesamiento en imágenes

Las técnicas de procesamiento en el dominio espacial actúan directamente sobre los píxeles de la imagen, trabajando en su entorno más cercano. Estas técnicas suelen emplear máscaras de forma cuadrada o rectangular, que son pequeñas matrices con valores predefinidos en cada una de sus posiciones (Sucar & Gómez, 2011). Por ejemplo, una máscara de 3 x 3 se utiliza para procesar la imagen considerando los píxeles vecinos de un píxel central. El nuevo valor de este píxel se determina en función de los valores de los píxeles que se encuentran bajo la máscara. El procedimiento de filtrado, también conocido como convolución, consiste en desplazar la máscara por toda la imagen, realizando operaciones específicas que permiten modificar las características visuales de

la misma, para entender mejor el procedimiento observemos la Figura 4, donde en ese caso se tiene como máscara un arreglo de 3x3 cuyos valores dentro de este representan a un filtro, el cual para iniciar la convolución se ubicó en uno de los pixeles de la imagen y la realizará una operación uno a uno con los valores que se encuentren en la misma posición (i,j)de la máscara para que finalmente el resultado se sume.

Matemáticamente se puede expresar lo anterior con la siguiente ecuación:

$$g(x,y) = \sum_i \sum_j f(i,j)\, w(i,j)$$

Donde:

$g(x,y)$: Valor del píxel en la posición (x, y) después de la convolución

$f(i,j)$: Valor del píxel antes de la convolución en la posición (i, j) de la máscara

$w(i,j)$: Peso en la posición (i, j) dentro de la máscara

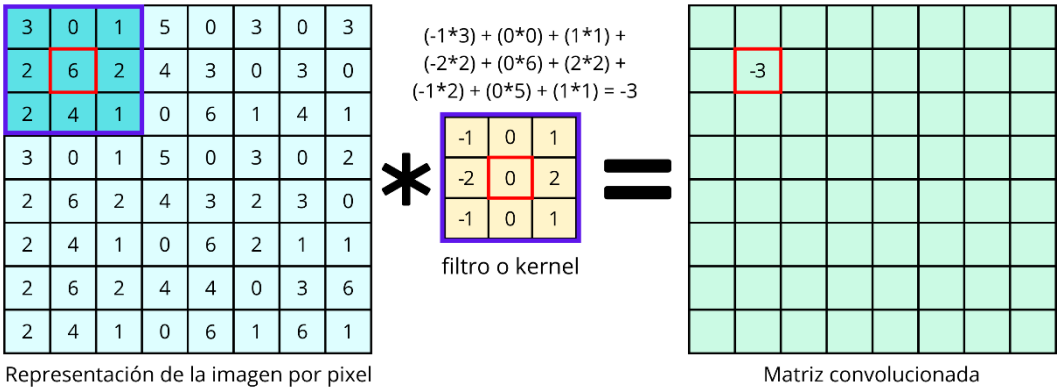


Figura 4: Ejemplo de una convolución empleando el filtro Sobel sobre una imagen. Adaptado de: Leppich (2021).

Por lo general, el resultado de la suma se divide entre un valor específico para normalizar. Esta máscara se aplica a cada píxel de la imagen, lo que implica realizar una operación de convolución entre la máscara y la imagen original. El tamaño de la máscara y los valores de sus coeficientes son los factores que determinan el tipo de filtrado que se llevará a cabo.

Dentro de las técnicas de procesamiento de imágenes encontramos al umbral de Otsu, el cual busca determinar de manera automática un umbral óptimo para la binarización de una

imagen. Este método, propuesto por Nobuyuki Otsu en 1979, se basa en la minimización de la varianza intra-clase, lo que permite dividir los píxeles en dos grupos bien diferenciados: fondo y objeto.

El fundamento del método de Otsu radica en analizar la distribución de niveles de intensidad de la imagen y encontrar el umbral que minimice la dispersión dentro de cada grupo. Para ello, se evalúa la varianza intra-clase $\sigma_W^2(T)$ definida como:

$$\sigma_W^2(T) = w_0(T)\sigma_0^2(T) + w_1(T)\sigma_1^2(T)$$

Donde:

$w_0(T),\ w_1(T)$: proporciones de píxeles en cada clase

$\sigma_0^2(T), \sigma_1^2(T)$: varianza de cada clase

El umbral óptimo T es aquel que minimiza esta varianza intra-clase, asegurando que los valores dentro de cada grupo sean lo más homogéneos posible.

Otra interpretación del método de Otsu es que maximiza la varianza inter-clase, que mide la separación entre los dos grupos de píxeles, definida como:

$$\sigma_B^2(T) = w_0(T)w_1(T)(\mu_0(T) - \mu_1(T))^2$$

Donde:

$\mu_0(T), \mu_1(T)$: medias de intensidad de cada grupo

El método de Otsu es utilizado en segmentación de imágenes, reconocimiento de patrones y visión por computadora, cuyo rendimiento óptimo se da en imágenes con distribuciones bimodales, donde los valores de los píxeles forman dos grupos bien definidos.

2.3.3.2 Técnicas de aumento de datos

El aumento de datos consiste en incrementar de forma artificial el tamaño del conjunto de entrenamiento mediante la generación de múltiples variaciones de cada muestra original.

Las transformaciones aplicadas permiten al modelo desarrollar una mayor tolerancia frente a variaciones en la posición, orientación y tamaño de los objetos presentes en las imágenes. Además, si se busca mejorar su capacidad para adaptarse a diferentes condiciones de iluminación, es posible generar imágenes con niveles de contraste variados.

Al combinar estas modificaciones, se amplía significativamente el conjunto de entrenamiento, lo que mejora la capacidad del modelo para generalizar y ofrecer un mejor rendimiento ante datos no vistos (Géron, 2019).

2.3.3.2.1 Rotación de imágenes para el aumento de datos

Las rotaciones de imágenes se logran mediante transformaciones geométricas que utilizan matrices de rotación aplicadas a los píxeles de la imagen. La rotación de un punto (x,y) alrededor del centro de la imagen (cx,cy) en un ángulo θ (en radianes) se expresa mediante la siguiente matriz de rotación:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -sen(\theta) \\ sen(\theta) & \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} x - cx \\ y - cy \end{bmatrix} + \begin{bmatrix} cx \\ cy \end{bmatrix}$$

Donde:

x', y': son las nuevas coordenadas del punto después de la rotación

θ: es el ángulo de rotación (positivo en sentido antihorario)

2.3.3.2.2 Cambios de brillo y contraste para el aumento de datos

El brillo y contraste dentro de una imagen se puede modificar ajustando los valores de los píxeles, la ecuación básica para ello es como la que se presenta a continuación:

$$I'(x,y) = \lambda \cdot I(x,y) + \beta$$

Donde:

I(x,y): intensidad original del píxel en la posición (x,y)

I’(x,y): intensidad luego del ajuste de brillo

λ: factor de contraste

λ =1: sin cambios en el contraste

λ >1: aumenta el contraste

λ =1: disminuye el contraste

β: factor de brillo

β=0: sin cambios en el brillo

β>0: aumenta el brillo

β<0: reduce el brillo

Si el resultado excede a 255, se satura a 255 si es menor que 0, se coloca 0; además en imágenes a color este ajuste se aplica a cada canal (R,G,B) por separado.

2.3.4 Redes neuronales convolucionales

Una red neuronal convolucional (CNN, por sus siglas en inglés, Convolutional Neural Network) es una red neuronal artificial diseñada para procesar y analizar datos, según LeCun et al. (2015) están diseñadas para procesar datos que se estructuran en forma de múltiples arreglos, como es el caso de una imagen en color, que se representa mediante tres matrices bidimensionales correspondientes a las intensidades de los píxeles en cada uno de los canales de color rojo, verde y azul).

La palabra convolucional vienes de la operación matemática “convolución” que permite aplicar un filtro o *kernel* sobre los datos para detectar patrones como bordes, texturas, colores, etc. Así también la palabra red neuronal, viene por la forma en la organización de las capas, similar a cómo funciona el cerebro humano para procesar información.

Gámiz (2019) presenta a la CNN como una composición de varias capas que trabajan en conjunto para extraer y procesar características de los datos de entrada, como se observa en la Figura 5, donde muestra parte de red convolucional que inicia con la matriz de entrada que representa la imagen original, donde cada píxel es procesado por filtros en la capa convolucional, generando mapas de características que resaltan patrones específicos. Posteriormente, en la capa de *pooling*, se reduce la dimensionalidad mediante operaciones como *max-pooling*, manteniendo las características más relevantes mientras se disminuye el tamaño de la representación. Este flujo refleja cómo la red abstrae progresivamente la información, pasando de detalles locales a representaciones más generales y compactas.

Matemáticamente, la operación de convolución de una CNN se define de la siguiente manera:

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_n'-1} x_{i',j',k'} \cdot w_{u,v,k',k}$$

Siendo:

$$\begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases}$$

Donde:

$z_{i,j,k}$: salida de la neurona ubicada en la fila i, columna j del mapa de características k de la capa convolucional l

s_h y s_w: pasos vertical y horizontal

f_h y f_w: altura y ancho del campo receptivo

f_n': número de mapas de características en la capa anterior (capa l-1)

$x_{i',j',k'}$: salida de la neurona localizada en la capa l-1, fila i colimna j del mapa de características k

b_k: término de bias para el mapa de características k (en la capa l)

$w_{u,v,k',k}$: pesos del filtro convolucional

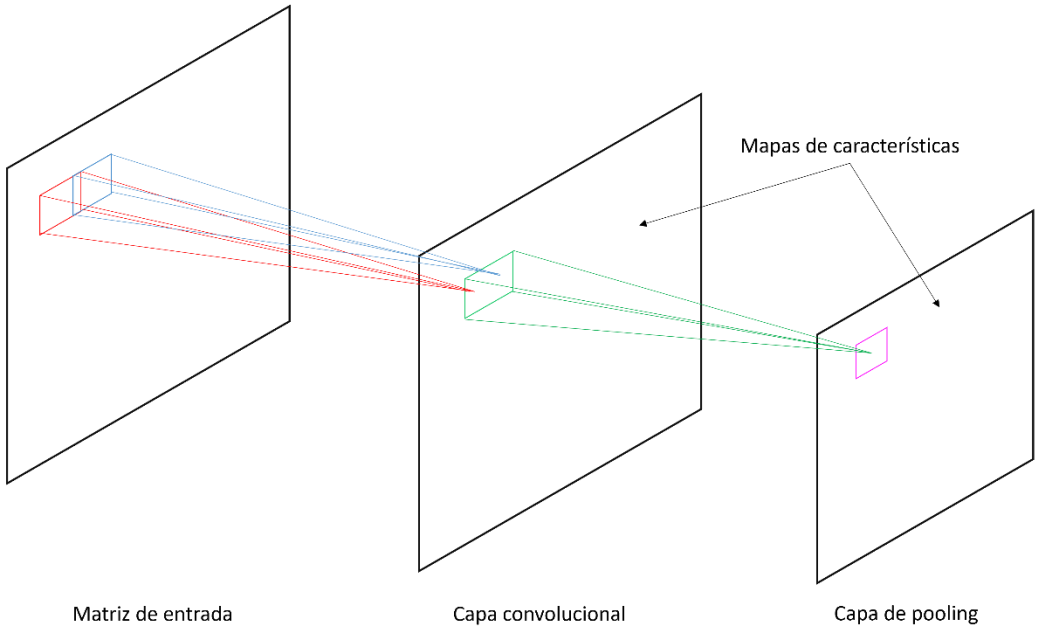


Figura 5: Parte de una red neuronal convolucional que contiene: capa convolucional y capa de pooling en campos receptivos rectangulares. Adaptado de Gámiz (2019).

La convolución es una operación que permite observar como una imagen o una función se influencia al interactuar con otra función, a la que comúnmente se le conoce como filtro o *kernel*. Esta interacción no ocurre de manera estática, ya que implica un proceso de deslizamiento en el cual el filtro se desplaza sobre la imagen o función, evaluando en cada posición la relación entre sus valores. La convolución, es una forma de recorrer e interpretar los datos, identificando patrones, bordes, texturas, etc.

2.3.5 Parámetros e hiperparámetros en redes neuronales

Los parámetros son valores internos del modelo que se aprenden automáticamente como los pesos, el *bias*, y los mapas de características. Por el contrario, los hiperparámetros son configuraciones establecidas antes del entrenamiento y no se actualizan automáticamente, entre ellos encontramos a las siguientes:

2.3.5.1 Optimizadores en deep learning

Según Goodfellow et al. (2016) un optimizador es un procedimiento utilizado para ajustar los parámetros de un modelo con la finalidad de minimizar una función de pérdida. Uno de los optimizadores más simples es el Descenso del Gradiente o *Gradient Descent*, que actualiza los pesos moviéndose en la dirección opuesta al gradiente de la función de pérdida para encontrar el mínimo. Una variante es el Descenso de Gradiente Estocástico (SGD), que realiza actualizaciones de los parámetros utilizando pequeños lotes de datos (*batch*), lo que acelera el proceso y mejora la generalización. También se tiene al, RMSProp que adapta la tasa de aprendizaje para cada parámetro individualmente. Por su parte, el optimizador Adam (Adaptive Moment Estimation) combina las ventajas de SGD con momentum y RMSProp, ya que ajusta la tasa de aprendizaje de forma adaptativa y basándose en dos momentos del gradiente durante el entrenamiento, el primer momento (m_t) que captura la dirección del gradiente a lo largo del tiempo, ayudando a suavizar las actualizaciones, y el segundo (v_t) que controla la magnitud de las actualizaciones ajustando la tasa de aprendizaje de forma adaptativa. Para inicializarse usa los valores de m_0, v_0 y t igual a cero y en cada iteración va actualizando el valor de la gradiente de la función de pérdida L con respecto a un parámetro θ en el paso t:

$$g_t = \nabla_{\theta_t} L(\theta_t)$$

Donde:

g_t: Gradiente actual

∇_{θ_t}: Derivada de la pérdida L con respecto a θ_t

Posteriormente se actualiza el valor del primer momento m_t y sel segundo momento v_t

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

Donde:

m_t : Momento de primer orden en el paso t

β_1 : Factor de decaimiento (usualmente es 0.9)

g_t : Gradiente actual

v_t : Momento de segundo orden en el paso t

β_2 : Factor de decaimiento (usualmente es0.999)

Inicialmente, m_t puede estar sesgado hacia cero. Para corregir este sesgo, Adam aplica una corrección.

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

Finamente, se actualizan los parámetros θ usando una tasa de aprendizaje adaptativa.

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\widehat{m}_t}{\sqrt{\hat{v}_t}+\in}$$

Donde:

θ_t : Parámetro actualizado en el paso t

α : Tasa de aprendizaje inicial (valor por defecto 0.001)

$\widehat{m}_t$: Momento de primer orden corregido

$\hat{v}_t$: Momento de segundo orden corregido

$\in$: Valor constante para evitar divisiones por cero (valor por defecto 10^{-8})

2.3.5.2 Funciones de pérdida en redes neuronales

Una función de pérdida es una función que asigna valores de una o más variables a un número real que representa un costo asociado a dicho evento, este costo refleja qué tan lejos está la predicción del modelo respecto al valor esperado. (Goodfellow et al, 2016). La

función de pérdida guía el proceso de aprendizaje proporcionando un valor que utiliza el optimizador Adam para minimizarlo, sin una función de pérdida bien definida la red neuronal no tendría forma de evaluar si está aprendiendo correctamente.

Entre las funciones de pérdida más comunes encontramos al error cuadrático medio que se utiliza en problemas de regresión, ya que calcula el promedio de los errores al cuadrado, penalizando con mayor severidad los errores grandes. Por su parte, el Error Absoluto Medio mide la magnitud del error promedio sin considerar su dirección, siendo más drástico frente a valores atípicos. En problemas de clasificación binaria, la Binary Cross-Entropy o llamada también *loss* en la práctica es la función más utilizada, ya que mide la pérdida basada en la probabilidad predicha para cada clase, penalizando fuertemente las predicciones incorrectas y guiando al modelo a mejorar su capacidad de distinguir entre dos categorías.

El cálculo de la función binary cross-entropy se realiza de la siguiente manera:

$$L = -\frac{1}{N}\sum_{i=1}^{N}(y_i \cdot \log(\hat{y}_i + \epsilon) + (1 - y_i) \cdot \log(1 - \hat{y}_i + \epsilon))$$

Donde:

L: Pérdida promedio

N : Números de muestras

y_i : Valor real de la muestra

$\hat{y}_i$: Probabilidad predicha por el modelo para la clase positiva

ϵ : Valor constante para evitar log(0)

Esta misma función se usa para calcular la pérdida en la validación (val_loss), su objetivo es evaluar la capacidad del modelo para generalizar, es decir, qué tan bien puede realizar predicciones precisas en datos que no ha visto antes. Mientras que el loss se calcula

utilizando el conjunto de entrenamiento, el *val_loss* se calcula utilizando un subconjunto separado de los datos, llamado conjunto de validación, que no interviene en el ajuste de los pesos del modelo y que se verá en títulos posteriores.

2.3.5.4 Tasa de aprendizaje

Es un hiperparámetro que determina el tamaño de los pasos que se dan hacia un mínimo de una función de pérdida durante el proceso de optimización (Goodfellow et al, 2016). Este controla el paso que da el optimizador al actualizar los parámetros del modelo, como los pesos y sesgos, en cada iteración. Un *learning rate* demasiado grande puede hacer que el modelo oscile sin llegar al óptimo y uno demasiado pequeño puede hacer que el modelo tarde mucho en converger o que se quede atrapado en un mínimo local.

2.3.5.5 División del conjunto de datos: entrenamiento y validación

Para evaluar cómo se desempeña un modelo predictivo, se suelen dividir el conjunto de datos en entrenamiento y prueba. Los datos de entrenamiento se utilizan para ajustar el modelo, permitiéndole aprender patrones y relaciones relevantes, mientras que los de prueba estiman la capacidad de generalización del modelo, es decir su habilidad para hacer predicciones en información nueva que no vió antes (Hastie et al., 2009)

Hastie et al. (2009) destacan que esta división ayuda a estimar el error de predicción y a prevenir el sobreajuste. Géron (2019) sugiere una proporción común de 80/20 entre entrenamiento y prueba, mientras que Goodfellow et al. (2016) enfatizan la importancia del conjunto de validación para ajustar los hiperparámetros sin introducir sesgos en la evaluación final.

2.3.5.6 Tamaño del lote y número de épocas

En el proceso de entrenamiento de redes neuronales, es importante comprender la relación entre iteración, tamaño de lote (batch size) y la época (epoch). Una iteración se refiere a una sola actualización de los parámetros del modelo, que ocurre después de procesar un

lote de datos. El tamaño de lote define cuántas muestras se incluyen en cada iteración, influyendo en la estabilidad y eficiencia del entrenamiento (Goodfellow et al., 2016), su uso permite dividir los datos en partes más pequeñas, de modo que el modelo procesa un batch a la vez en lugar de todos los datos simultáneamente, lo que reduce el costo computacional. A su vez, una época representa un ciclo completo en el que el modelo ha visto todas las muestras del conjunto de entrenamiento al menos una vez, lo que implica múltiples iteraciones dependiendo del tamaño del lote (Géron, 2019).

La relación entre estos conceptos viene dada por la siguiente ecuación:

$$Número\ de\ iteraciones\ por\ época = \frac{Tamaño\ total\ del\ conjunto\ de\ datos}{Tamaño\ del\ lote}$$

2.3.5.7 Callbacks

Un callback es una función especial que se ejecuta automáticamente durante el entrenamiento de un modelo para controlar su comportamiento. Estos controlan el proceso de manera dinámica permitiendo a los usuarios detener el entrenamiento antes de que culmine con todas épocas, guardar los modelos y ajustar hiperparámetros en respuesta al rendimiento de la validación (Chollet, 2017).

Entre los callbacks más empleados, se encuentra Early Stopping, una técnica utilizada para detener automáticamente el entrenamiento de un modelo cuando su desempeño en el conjunto de validación deja de mejorar. Su principal objetivo es evitar el sobreajuste, asegurando que el modelo no siga entrenando innecesariamente una vez que ha alcanzado su mejor rendimiento, evitando así también el sobreentrenamiento (overfitting). Este callback monitorea una métrica específica, como la pérdida de validación o la precisión de validación, y si esta no mejora después de un número determinado de épocas, el entrenamiento se interrumpe. Además, suele incluir la opción de restaurar los pesos del modelo al mejor estado alcanzado, garantizando que se utilice la versión óptima del modelo en lugar de aquella en la que comenzó a deteriorarse su rendimiento.

El comportamiento de este *callback* está controlado principalmente por dos parámetros: *patience*, que define en número de épocas consecutivas permitidas sin mejora en la métrica monitoreada antes de detener el entrenamiento y min_delta, que establece la mejora mínima requerida para considerar que hubo progreso en la métrica monitoreada, deteniendo el entrenamiento si no se observa una mejora significativa determinada en el parámetro “min_delta” como se indica en las ecuaciones siguientes, donde de no cumplirse dichas condiciones un número de épocas consecutivas, el entrenamiento se detiene.

$$val\_loss_t < val\_loss_{t-1} - min\_delta$$

$$val\_accuracy_{t-1} + min\_delta < val\_accuracy_t$$

Otro de los *callbacks* más empleados es el CSVLogger, una herramienta utilizada para registrar en un archivo CSV el historial de entrenamiento del modelo, incluyendo las métricas de entrenamiento, permitiendo el análisis posterior del modelo después del entrenamiento, ya que permite almacenar los resultados de cada época facilitando su posterior visualización o análisis con herramientas como Excel, Pandas o *Matplotlib*.

2.3.6 Evaluación del modelo: métricas de desempeño

Las métricas en el aprendizaje automático son funciones diseñadas para evaluar el rendimiento de un modelo, permitiendo medir la calidad de sus predicciones y determinar qué tan bien está cumpliendo con su objetivo. Estas permiten comparar diferentes modelos, ajustar hiperparámetros y monitorear el progreso durante el entrenamiento. Goodfellow, Bengio y Courville (2016) destacan que, si bien la función de pérdida guía el proceso de optimización del modelo, las métricas proporcionan medidas interpretables del rendimiento, ayudando a los a tomar decisiones informadas sobre la eficacia del modelo en tareas específicas.

En tareas de segmentación, las métricas de evaluación miden la precisión con la que un modelo identifica y delimita regiones de interés en una imagen. Algunas métricas utilizadas en clasificación también se aplican en segmentación, adaptadas al contexto de

predicciones píxel a píxel, entre ellas tenemos a la precisión (*precision*) calcula el porcentaje de píxeles correctamente clasificados en toda la imagen, aunque puede ser engañosa si hay un desbalance entre clases, el ratio de verdaderos positivos (*recall*) que mide la capacidad del modelo para detectar todos los píxeles positivos reales, el F1-score combina precisión y exhaustividad en una métrica balanceada y el IoU (Intersection over Union) mide la superposición entre la máscara predicha y la máscara real, evaluando qué tan bien coincide la segmentación del modelo con el área de interés real. El F1-score y el IoU son particularmente relevantes en segmentación, por lo que los detallaremos a continuación.

2.3.6.1 Métrica F1-score

Esta métrica mide el rendimiento de un modelo de aprendizaje automático, ofreciendo un equilibrio entre *precision* y *recall*, penalizando los modelos que favorecen una de estas métricas en perjuicio de la otra. Un valor de 1 indica un rendimiento perfecto, mientras que un valor de 0 refleja un desempeño deficiente.

Se define como la media armónica entre la precisión (*precision*), proporción de píxeles correctamente clasificados como positivos entre todos los clasificados como positivos, y la exhaustividad (*recall*), que mide la proporción de píxeles correctamente clasificados como positivos entre todos los que son realmente positivos, la Figura 6 muestran el cálculo de estas de una manera más didáctica para una mejor comprensión.

El cálculo de F1-Score viene dado por la siguiente ecuación:

$$F1 - Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

ó

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

Donde:

TP: Verdaderos positivos

FP: falsos positivos

FN: falsos negativos

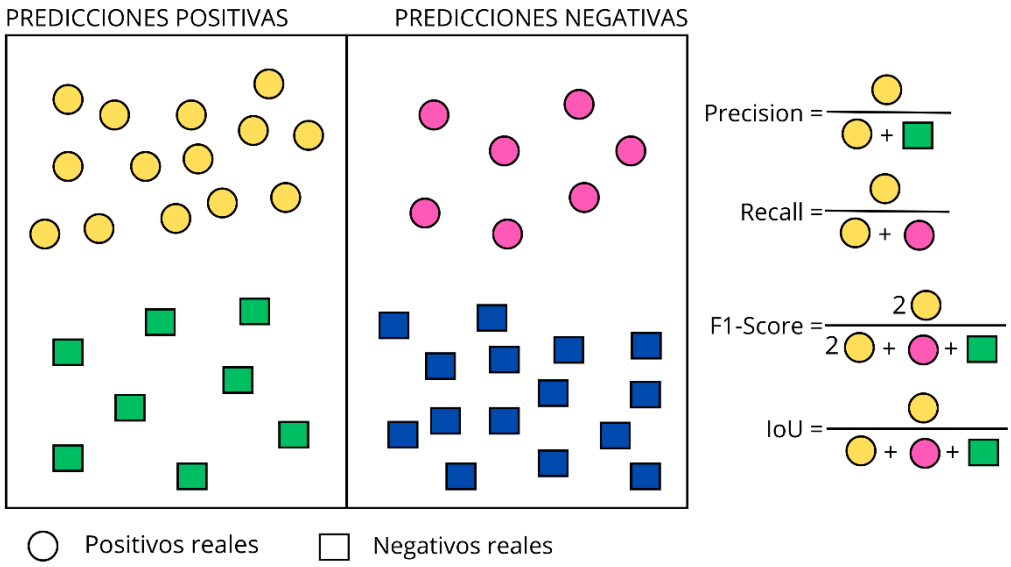


Figura 6: Cálculo de precision, Recall, F1-Score e IoU.

2.3.6.2 Métrica intersección sobre la unión promedio

Esta métrica mide la intersección entre la máscara predicha y la máscara real, proporcionando una forma intuitiva de evaluar qué tan bien coinciden las áreas segmentadas por el modelo con las regiones reales de interés, un valor de 1 indica un rendimiento perfecto, mientras que un valor de 0 refleja un desempeño deficiente.

Este coeficiente se puede calcular de la siguiente manera:

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

Donde:

TP: Verdaderos positivos

FP: falsos positivos

FN: falsos negativos

Aunque el IoU y el F1-score son métricas de superposición, no son exactamente iguales, pero están estrechamente relacionados como se muestra en la siguiente ecuación:

$$IoU = \frac{(F1 - Score)}{2 - (F1 - Score)}$$

2.3.7 Modelos y arquitecturas de Deep Learning para segmentación

Un modelo en *deep learning* es la implementación de una arquitectura de red neuronal que ha sido entrenada con datos para realizar tareas específicas, un modelo no solo incluye la arquitectura, sino que también los parámetros aprendidos como los pesos y bias. (Goodfellow et al, 2016). Este puede iniciar su proceso de aprendizaje con pesos y parámetros inicializados aleatoriamente, al que se le llamaría modelo sin preentrenamiento o puede iniciar con datos ya aprendidos (modelo preentrenado) ya que ha sido previamente entrenado con un conjunto de datos, ahorrando tiempo y recursos computacionales.

Una arquitectura se refiere a la estructura y organización de una red neuronal, es decir, la disposición de sus capas, el tipo de conexiones entre ellas y cómo fluye la información desde la entrada hasta la salida del modelo. La arquitectura es la transformación de los datos a través de múltiples capas extrayendo características de ellas para hacer una predicción. (Goodfellow et al, 2016).

En segmentación de imágenes, se han desarrollado diversas arquitecturas para la segmentación, donde es necesario la clasificación por píxel de una imagen. Entre las más destacadas se encuentra U-Net, propuesta por Ronneberger et al. (2015), el cual cuenta con un diseño en forma de "U" con un codificador (encoder) para extraer características y un decodificador (decoder) para reconstruir la segmentación, siendo efectiva en aplicaciones biomédicas. Otra arquitectura relevante es DeepLabV3+ (Chen et al., 2018), que utiliza convoluciones dilatadas y el módulo *Atrous Spatial Pyramid Pooling* (ASPP) para capturar información contextual en diferentes escalas, ideal para segmentación

semántica compleja. Además, *Fully Convolutional Network* (FCN), introducida por Long et al. (2015), fue la primera red completamente convolucional diseñada para segmentación, reemplazando las capas densas por convoluciones que permiten procesar imágenes de cualquier tamaño. Otras arquitecturas como ResNet (He et al., 2016) y VGG (Simonyan & Zisserman, 2014) se utilizan como extractores de características (*backbones*) en modelos de segmentación más complejos, debido a su capacidad para aprender representaciones profundas y jerárquicas.

2.3.7.1 Arquitectura U-Net

U-Net es una arquitectura de red neuronal convolucional que fue diseñada para tareas de segmentación de imágenes biomédicas. Fue introducida por Ronneberger en el año 2015 y su estructura se caracteriza por una forma de "U" que combina un camino de contracción (*encoder*) y uno de expansión (*decoder*) conectados mediante *skip connections*, junto con la integración de características en diferentes niveles de resolución, resultando en una segmentación precisa incluso en estructuras complejas.

El *encoder* se encarga de extraer características relevantes de la imagen de entrada, mediante la aplicación repetida de filtros, cada una seguida por una función de activación de unidad lineal rectificada (*ReLU* por sus siglas en inglés, rectified linear activation function), que introduce no linealidad para aprender patrones complejos. Posteriormente se utiliza una operación de *max-Pooling* para reducir las dimensiones espaciales, lo que disminuye la complejidad computacional y concentra las características más abstractas.

El *decoder* reconstruye la imagen segmentada a partir de las características extraídas en el codificador, mediante operaciones de *upsampling* que amplían la resolución de la imagen a través de una convolución transpuesta, que permite duplicar la resolución espacial mientras aprende patrones adaptativos. Posteriormente las características resultantes se refinan a través de convoluciones adicionales y la concatenación con los mapas de características correspondientes del *encoder* mediante conexiones de salto (*skip*

connections) para que así se pueda recuperar detalles perdidos durante la compresión. Finalmente, se aplican funciones de activación *ReLu* para mejorar la no linealidad y la capacidad del modelo de capturar patrones complejos, contribuyendo a una reconstrucción de la imagen. A continuación, en la Figura 7, se muestra el proceso de la arquitectura U-Net descrito anteriormente.

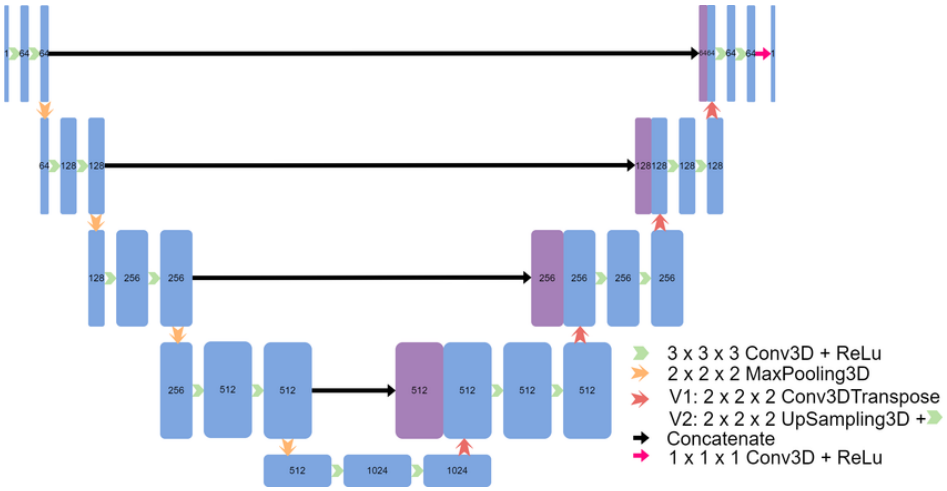


Figura 7: Diagrama de la arquitectura U-Net. Fuente: Tomado de Munro et al. (2023).

2.3.7.2 Shallow U-Net

El término shallow, significa poco profundo por lo que esta arquitectura vendría a ser una variante más simple de U-Net que conserva la estructura en forma de “U” pero con menos características, es decir con menos capas en los caminos de contracción y expansión. Esta arquitectura requiere menos recursos computacionales y además puede converger más rápido.

2.3.7.3 ResNet101 U-Net

ResNet (*Residual Network*) es una arquitectura de red neuronal profunda diseñada para facilitar el entrenamiento de redes extremadamente profundas mediante el uso de conexiones residuales (*skip connections*). Estas conexiones permiten que el flujo de información y los gradientes pasen directamente a través de la red, lo que evita problemas como el desvanecimiento del gradiente en redes muy profundas. En lugar de aprender directamente la función objetivo, la red aprende la diferencia (residuo) entre la entrada y la

salida esperada, lo que simplifica la optimización y mejora la eficiencia del aprendizaje. Gracias a este enfoque, ResNet permite construir redes con más de 100 capas, como ResNet50 o ResNet101, sin que el rendimiento se degrade (He et al., 2016).

Por su parte, ResNet101 U-Net es una arquitectura híbrida que combina la potencia de ResNet101 como *backbone* y codificador (*encoder*) para la extracción de características complejas, con la estructura de U-Net, optimizada para la segmentación de imágenes. En esta configuración, ResNet101 procesa la imagen de entrada a través de sus bloques residuales, generando mapas de características a diferentes niveles de resolución. Estos mapas se integran en el decodificador (*decoder*) de U-Net mediante conexiones de salto, lo que permite incorporar información detallada durante las etapas de *upsampling* y convolución del decodificador, mejorando la precisión de la segmentación final.

2.3.7.4 VGG U-Net

Visual Geometry Group (VGG) es una arquitectura de red neuronal convolucional desarrollada por Simonyan & Zisserman (2015), conocida por su diseño simple y basado en el uso de múltiples capas convolucionales secuenciales, seguidas de capas de *max-pooling* para la reducción de la dimensión espacial. Esta estructura permite que la red aprenda representaciones jerárquicas de las imágenes, desde patrones simples como bordes hasta características complejas, a través de su profundidad combinada con filtros pequeños. (Simonyan & Zisserman, 2015)

VGG U-Net es una arquitectura híbrida que combina dos redes neuronales VGG que se encarga de extraer características profundas (encoder) y U-Net que reconstruye la imagen segmentada a partir de las características extraídas, en la Figura 8 podemos observar cómo se distribuyen las capas en la arquitectura, en el encoder, donde la imagen de entrada pasa por varias capas convolucionales 3x3 con ReLU, seguidas de capas de MaxPooling (2x2) que reducen la resolución mientras aumentan la cantidad de filtros. En el decoder, la información se reconstruye progresivamente mediante upsampling y

convoluciones adicionales, restaurando la resolución original de la imagen. Además, las skip connections conectan las capas equivalentes del encoder y el decoder, permitiendo recuperar detalles espaciales perdidos en la compresión. Finalmente, la salida es una máscara de segmentación, donde cada píxel recibe una etiqueta según su clase correspondiente.

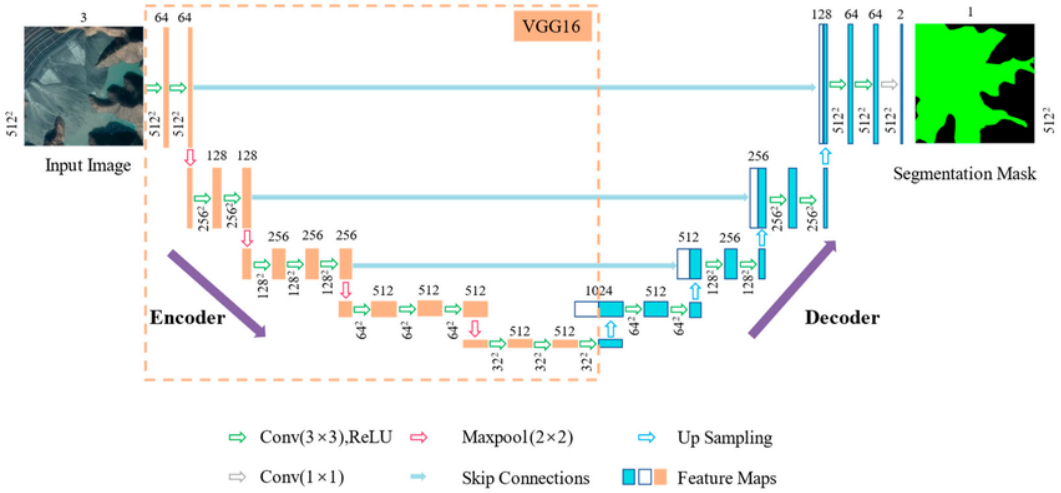


Figura 8: Diagrama de la arquitectura VGG16 U-Net. Fuente: Tomado de Wang (2023)

2.3.7.5 DeepLabv3+

Es una arquitectura desarrollada para mejorar la delineación de objetos en sus bordes. Esta técnica combina convoluciones dilatadas y módulos de codificación-decodificación para capturar información contextual a múltiples escalas y refinar los detalles espaciales en las predicciones de segmentación. (Chen et al, 2018).

Esta arquitectura está compuesta por tres componentes importantes que son el backbone que se encarga de extraer características relevantes de las imágenes entrada a través de modelos preentrenados como ResNet50, ResNet101 o Xception. Luego pasa los mapas de características al módulo ASPP que captura información de objetos de diferentes tamaños de la misma imagen, usa convoluciones dilatadas con diferentes tasas de dilatación para aumentar su campo receptivo sin perder resolución espacial y genera un mapa de características enriquecido con información global y detalles locales, resultando en una combinación de características de distintos niveles. Finalmente añade un

decodificador que mejora la segmentación recuperando detalles perdidos en el encoding y el upsampling, fusiona la información de ASPP con las características originales de la imagen para obtener una segmentación más precisa.

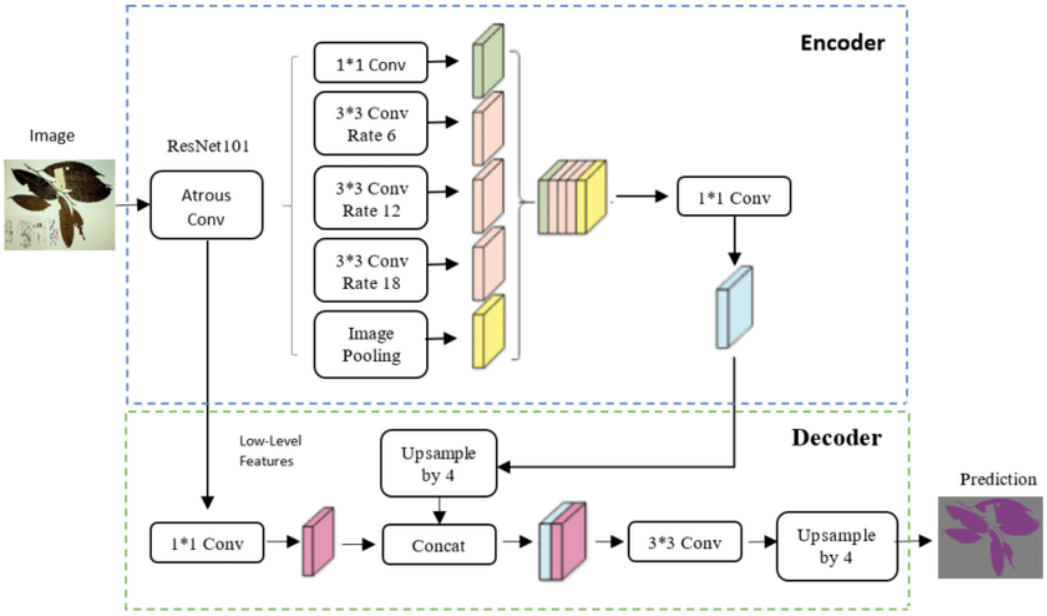


Figura 9: Diagrama de la arquitectura DeepLabv3+. Fuente: Tomado de Hussein et al. (2020).

En la Figura 9 se muestra la representación de la arquitectura de DeepLabV3+, que muestra en la parte superior (*encoder*), donde la imagen de entrada es procesada por el *backbone* Resnet101 aplicando convoluciones dilatadas en las capas profundas. Luego, la información es procesada a través del módulo, el cual aplica convoluciones de diferentes tasas de dilatación (*Rate* 6, 12 y 18) y una capa de *Image Pooling*, lo que permite capturar información en múltiples escalas. Finalmente, se realiza una convolución 1x1 que reducir la dimensionalidad de los canales sin perder información. En la parte inferior (*decoder*), se combinan las características de bajo nivel provenientes del *encoder* mediante una convolución 1x1 y un *upsampling* por un factor de 4. Luego, estas características se concatenan con las del *encoder* y se procesan con una convolución 3x3 para refinar la segmentación. Finalmente, un segundo *upsampling* por 4 reconstruye la imagen segmentada en su resolución original, generando la predicción final con las regiones segmentadas correctamente identificadas.

Capítulo III. Preprocesamiento de imágenes de vuelos históricos

Antes de detallar el preprocesamiento de imágenes de vuelos históricos, es importante aclarar que, en este estudio, los términos fallas y grietas se utilizarán de manera indistinta, ya que las grietas representan una de las formas más comunes de deterioro en pavimentos. Según el Manual del PCI, que establece los criterios para la evaluación de fallas en pavimentos, existen seis tipos de grietas: piel de cocodrilo, agrietamiento en bloque, grieta de borde, grieta de reflexión de junta, grietas parabólicas y grietas longitudinales y transversales. En esta investigación, no se hará distinción entre estas categorías, priorizando su detección y segmentación de manera integral.

El proceso de construcción de la base de datos se presenta en la Figura 10, mediante un diagrama de flujo que muestra las etapas clave involucradas. Este proceso abarca desde las dos fuentes principales de datos, los ortomosaicos históricos y las fotografías capturadas con cámara que complementarán esta base de datos, generando finalmente una base de datos preparada para el entrenamiento del modelo de segmentación.

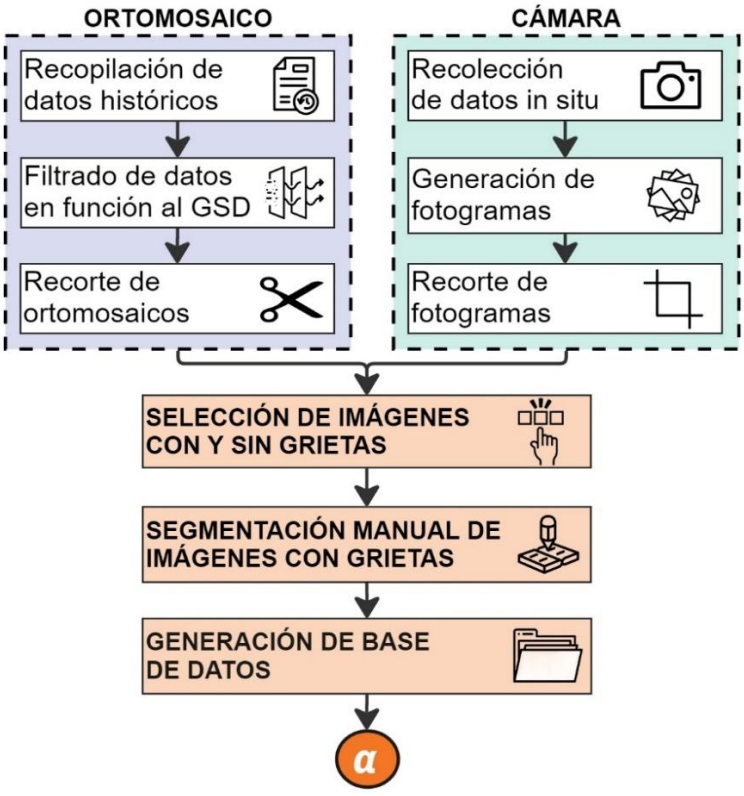


Figura 10: Diagrama del proceso de generación de la base de datos.

3.1 Metodología de recolección de datos

La recolección de datos se basó en dos principales fuentes de información. La primera correspondió a ortomosaicos, cuyos recortes fueron utilizados como datos de entrenamiento debido a su similitud con los patrones típicos de entrada, que también serán conformados por ortomosaicos. La segunda fuente consistió en imágenes capturadas con una cámara de alta resolución, la cual, gracias a la proximidad al pavimento, permitió obtener imágenes de mayor calidad, facilitando así la identificación de grietas de menor ancho con mayor detalle.

3.1.1 Recolección de imágenes a través de ortomosaicos

El CISMID (Centro Peruano Japonés de investigaciones sísmicas y mitigación de desastres) cuenta con datos históricos de vuelos realizados con drones en diversos distritos del Perú, principalmente en distritos de la ciudad de Lima. Estos vuelos han permitido generar ortomosaicos ortorectificados, los cuales podrían ser empleados como datos de entrenamiento tras un filtrado.

La recolección de imágenes para la generación de los ortomosaicos se llevó a cabo utilizando un dron Mavic 2 pro en los distritos listados en la Tabla 1, exceptuando el vuelo realizado en la Av. Julio César Tello, en el distrito de Ancón, que fue ejecutado con un dron Mavic 3 Entreprise. Aunque no se dispone de información específica sobre los parámetros de cada vuelo, como altura de vuelo, traslape, o velocidad, se logró obtener el GSD a partir de la metadata presente en cada uno de los ortomosaicos. Este parámetro es importante ya que permite estimar la resolución del ortomosaico y será importante para seleccionar los datos más adecuados para el entrenamiento del modelo.

Tabla 1: Características de los ortomosaicos de la base de datos histórica del CISMID.

ID	Distrito - Referencia	Tamaño Ortomosaico (píxel x píxel)	GSD (cm/píxel)
1	Ancón - Playa Pocitos	8,970 x 17,169	4.23
2	Ancón - Av. Julio César Tello[1]	102,086 x 85,926	1.51
3	Ancón - Urbanización Miramar	20,987 x 43,743	2.58
4	Chorrillos - Costa Negra	32,128 x 18,805	1.62
5	Coishco - Vuelo 1	22,198 x 8,578	1.81
6	Coishco - Vuelo 2	20,125 x 7,812	2.00
7	Coishco - Vuelo 3	25,952 x 13,291	1.75
8	Coishco - Vuelo 4	31,484 x 13,085	2.10
9	Coishco - Vuelo 5	10,467 x 14,021	2.10
10	Coishco - Vuelo 6	13,978 x 20,758	1.72
11	Coishco - Vuelo 7	15,221 x 16,161	1.75
12	Comas - Av. Maestro Peruano 1	47,165 x 16,252	1.41
13	Comas - Av. Maestro Peruano 2	25,696 x 9,747	1.40
14	Comas - Av. Maestro Peruano 3	35,102 x 20,599	1.39
15	Comas - A.H Vista Alegre	11,298 x 12,170	4.79
16	Callao - La Punta	26,801 x 26,880	2.54
17	Lima - Barrios Altos	20,545 x 44,297	1.93
18	Lima - Hospital Loayza	4,792 x 9,404	2.60
19	Pueblo Libre - Bandera	127,181 x 16,984	2.13
20	Pueblo Libre - Mariano Cornejo	125,372 x 16,734	2.16
21	Rímac - UNI	33,026 x 76,007	2.44
22	San Isidro - Prescott & Javier Prado	21,342 x 16,055	1.57
23	San Isidro - Salaverry & Pershing	19,674 x 20,840	1.81
24	San Isidro - Acantilado Costa Verde	22,746 x 20,230	5.00
25	San Martín de Porres - Habich	102,212 x 71,693	0.52
26	Surco - Primavera60m	24,788 x 13,372	1.59
27	Surco - Primavera80m	57,489 x 40,000	2.23
28	Surco - Primavera100m	44,725 x 32,986	2.87

(1) Ortomosaico levantado con el dron MAVIC 3E.

3.1.1.1 Características de los drones

Las características de los drones empleados en esta investigación se presentan en la Tabla 2, donde se muestra que el dron Mavic 2 Pro está equipado con una cámara integrada que utiliza un sensor de 1” CMOS (Semiconductor complementario de óxido metálico), lo que permite capturar imágenes de buena calidad y con un amplio rango dinámico. La cámara cuenta con una distancia focal de 9 mm y genera imágenes con una resolución de 5472 x 3648 píxeles.

Por otro lado, el dron Mavic 3 Enterprise está equipado con una cámara que posee un sensor de 4/3 CMOS y una distancia focal de 12 mm, generando imágenes con una resolución de 5472 x 3648 píxeles (Da Jiang Innovations [DJI], 2024), además cuenta con un módulo RTK integrado, permitiendo capturar imágenes con precisión centimétrica en tiempo real.

Ambos drones están equipados con tecnología de estabilización en 3 ejes, lo que garantiza la captura de imágenes nítidas y detalladas, incluso en vuelos inestables.

Tabla 2: Resumen de las características de los drones utilizados

Características	**Mavic 2 Pro**	**Mavic 3 Enterprise**
Tamaño del sensor (CMOS)	1”	4/3
Distancia focal de la cámara (mm)	9	12
Resolución de las imágenes (pixeles)	5472 x 3648	5472 x 3648
RTK integrado	No	Sí

3.1.2 Recolección de imágenes con cámara de alta resolución

Un estudio realizado por Mei & Gül (2019) determinó que las cámaras deportivas montadas en vehículos en movimiento, como las GoPro, son una herramienta efectiva para la inspección de grietas en carreteras, destacándose por su capacidad para detectar grietas con precisión en el pavimento. Por esta razón, en esta investigación se decidió incorporar el uso de cámaras de alta resolución, para que complemente las imágenes obtenidas del

ortomosaico, proporcionando un mayor nivel de detalle en las grietas presentes en el pavimento.

La recolección de imágenes con estas cámaras se realizó utilizando un vehículo mediano (miniván), lo que permitió cubrir áreas extensas en un menor tiempo. Para este propósito, se realizó una grabación de video del pavimento flexible en la Av. Julio César Tello, ubicada en el distrito de Ancón, en la ciudad de Lima, entre las 4:00 y 5:00 pm. Este horario fue elegido debido a las condiciones favorables de iluminación y tráfico, lo que garantizó una captura óptima de los datos. La elección de esta avenida se debió al avanzado estado de deterioro de su pavimento, lo que la convertía en un escenario ideal para generar una base de datos amplia y variada, adecuada para el entrenamiento del modelo de segmentación, ya que proporciona ejemplos representativos de diferentes tipos y tamaños de grietas.

Las grabaciones se realizaron con la ayuda de una varilla de extensión de aproximadamente 50 cm conectada a la cámara. Esta configuración permitió capturar aproximadamente la mitad del ancho del carril, no interferir con el tránsito vehicular y evitar las sombras proyectadas por el vehículo, un esquema a de este proceso se muestra en la Figura 11.

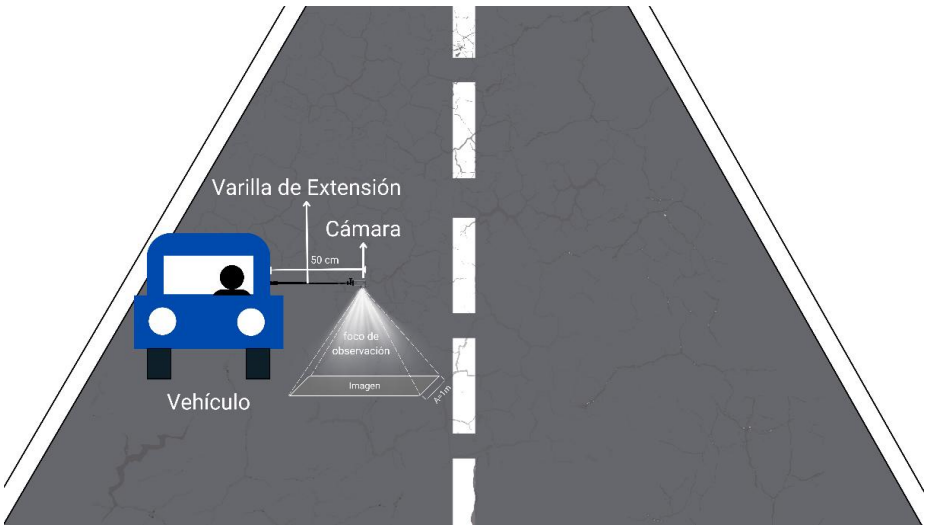


Figura 11: Esquema de toma de datos en vías utilizando una cámara de alta resolución.

La cámara se montó en un plano paralelo al de la vía para capturar imágenes del pavimento y reducir las distorsiones en el video. Esta forma de toma de datos minimizó los riesgos

asociados a caminar en zonas de tráfico vehicular, pues según el artículo "Pedestrian Safety" publicado en la revista científica Pediatrics, la seguridad peatonal tiene altos riesgos de accidentes en zonas de tráfico denso, por lo que al utilizar un vehículo se obtuvo una mayor seguridad durante el proceso de toma de datos (Committee on Injury, Violence, and Poison Prevention, 2009).

3.1.1.1 Características de la cámara de alta resolución

Las grabaciones se realizaron utilizando una cámara de acción DJI Osmo Action 4 Adventure, cuyo peso es de 145 gramos, brindando así versatilidad, maniobrabilidad y comodidad, permitiendo realizar grabaciones prolongadas sin causar fatiga. La cámara cuenta con una resolución de 1920x1080 píxeles, una tasa de hasta 120 fotogramas por segundo (fps), una distancia focal mínima de 40cm y una altura del sensor de 2cm, que permite capturar imágenes nítidas y bien enfocada a distancias cortas. Esta cámara también está equipada con estabilización de imagen, lo que minimiza las vibraciones y asegura una buena calidad de grabación en entornos dinámicos (DJI, 2024).

3.2 Procesamiento y categorización de imágenes recopiladas

A partir de los datos recolectados, se realizaron recortes a las imágenes, las cuales fueron clasificadas en dos categorías: imágenes con grietas e imágenes sin grietas. Además, se aplicaron criterios adicionales para descartar ciertas imágenes, las cuales se detallarán en las siguientes secciones.

3.2.1 Procesamiento de imágenes para el ajuste dimensional

Para la presente investigación se consideraron imágenes con dimensiones de 448x448 pixeles para el entrenamiento del modelo. Esta elección se realizó con el objetivo de evitar el uso de imágenes de gran tamaño, optimizando la capacidad de procesamiento del modelo. Al mismo tiempo, se aseguró que no se pierda información crítica sobre las grietas, manteniendo una resolución adecuada.

Esta decisión está alineada con investigaciones recientes, como la de Yang & Li (2020), quienes utilizaron imágenes con dimensiones para entrenar un modelo de detección de grietas en estructuras de concreto armado mediante el algoritmo YOLO, logrando una detección precisa. Del mismo modo, Ma (2021) utilizó imágenes del mismo tamaño para la segmentación de grietas en el pavimento rígido, apoyándose en la red neuronal Resnet101, donde el algoritmo demostró una alta precisión en la detección y una buena capacidad de generalización. Además, Zhang et al. (2024) utilizaron imágenes de 448x448 pixeles para entrenar la arquitectura CrackDiff, obteniendo un buen rendimiento para su modelo, lo que subraya la eficacia de este tamaño de imagen en la detección y segmentación de grietas.

Por ello, en los siguientes apartados se detallará el proceso de ajuste dimensional de los datos recopilados, tanto de los ortomosaicos como de las imágenes de video capturadas con cámara. Esto fue necesario debido a que el tamaño original de estos datos superaba las dimensiones establecidas para los datos de entrenamiento.

3.2.1.1 Ajuste dimensional del ortomosaico

Previo al ajuste dimensional del ortomosaico a través de recortes, se realizó un análisis visual de las imágenes basado en el GSD, con el objetivo de delimitar el área de estudio a aquellas grietas que conserven sus características visuales ya que un menor valor de GSD proporciona mayor detalle en las imágenes, lo que resulta importante tanto para preservar la calidad de la información sobre las grietas como para facilitar la segmentación manual, que se detallará en títulos posteriores. Es importante destacar que no se encontraron investigaciones previas que analicen específicamente la relación entre la pérdida de datos durante el entrenamiento de modelos de inteligencia artificial y el GSD, esto abre la posibilidad de realizar futuras investigaciones sobre el tema.

Para afrontar esta limitación, la presente investigación realizó un análisis cualitativo de las imágenes, en el cual, mediante una inspección visual, se determinó si las grietas presentes

en la fotografía aún conservaban sus características o, por el contrario, si se confundían con el pavimento. Como punto de partida, se seleccionó un ortomosaico de los vuelos históricos disponibles, el cual tenía un GSD DE 0.54 cm/pixel, considerado como el valor mínimo para este trabajo ya que corresponde al mínimo que se puede obtener utilizando los drones especificados, volando a una altura de 20 metros, que es la mínima permitida para vuelos de drones según la Norma Técnica Complementaria NTC-001-2015 (Dirección General de Aeronáutica Civil, 2015). Por lo tanto, este GSD se definió como el valor de referencia mínimo en la presente investigación.

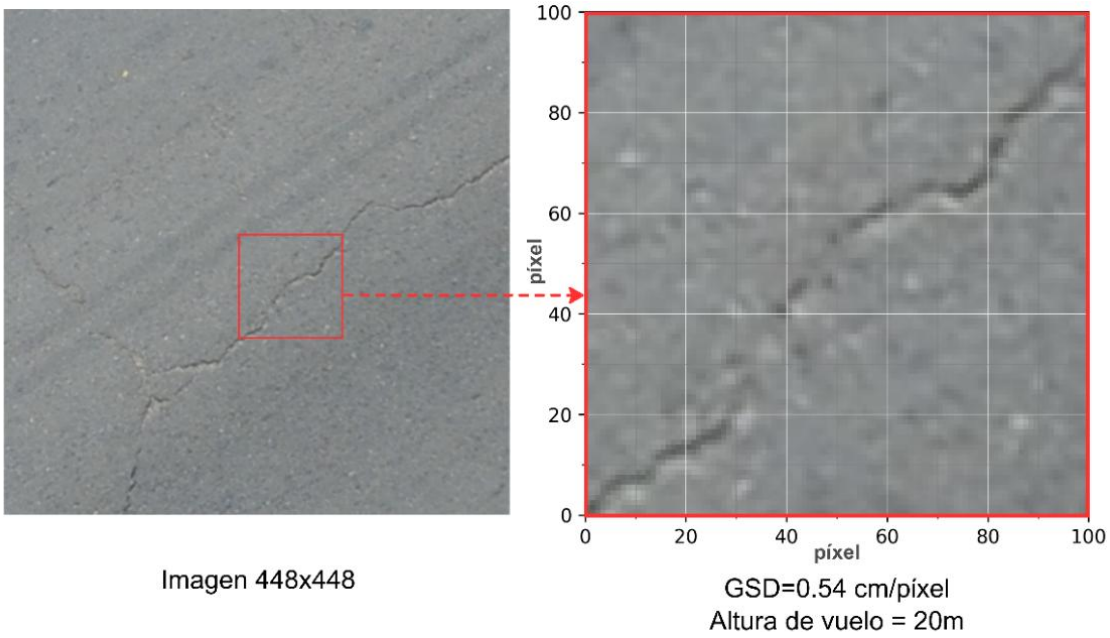


Figura 12: Imagen base y su extracción de muestra de 100x100 pixeles.

A partir de este ortomosaico, se recortó una porción de imagen, en tamaño 448x448, esta porción contenía una grieta dentro de ella y se denominó imagen base. De la imagen base, se extrajo un sector de 100x100 píxeles, cuidando que la grieta se encuentre dentro de este sector, como se muestra en la Figura 12. Posteriormente, se simularon vuelos a las alturas de 30 m, 40 m, 50 m, 60 m, 70 m y 80 m, calculando el GSD correspondiente para cada altura.

Se consideraron los parámetros técnicos del dron descritos en títulos anteriores y se realizó un redimensionamiento del sector recortado, ajustando el tamaño de la imagen de acuerdo con el GSD obtenido. Los resultados de los redimensionamientos para alturas mayores a

20 m se presentan en la Figura 13, mientras que los tamaños correspondientes de las imágenes redimensionadas se detallan en la Tabla 3.

Tabla 3: Redimensionado de imágenes para la simulación de alturas de vuelo.

Altura (m)	GSD (cm/píxel)	Tamaño de imagen redimensionada (píxel x píxel)
20	0.54	100 x 100
30	0.80	67 x 67
40	1.07	50 x 50
50	1.34	40 x 40
60	1.61	34 x 34
70	1.88	29 x 29
80	2.14	25 x 25

En la Figura 14 se observa que, a medida que aumenta la altura de vuelo, los píxeles pierden calidad. Esto se debe a que, al analizar las etiquetas en los ejes que indican la cantidad de pixeles, estas ya no conservan el valor inicial de 100x100 si no que disminuyen conforme aumenta el GSD. En otras palabras, para un mismo dron, el GSD varía con la altura lo que indica que cada píxel cubra una mayor área. Por lo tanto, se requieren menos píxeles para cubrir la misma superficie, lo que resulta en una pérdida de detalles a medida que disminuye la cantidad de píxeles disponibles.

En base a este análisis se determinó que la altura máxima a la que las grietas aún conservan sus detalles se encuentra entre 50m y 60m, por lo que escogió un GSD promedio de 1.47 cm/píxel. A una altura de 60m, las grietas, ya comienzan a difuminarse, como se observa en el área encerrada en el círculo amarillo de la Figura 13, perdiendo detalles importantes. Considerando los vuelos históricos disponibles, solo se tomarán en cuenta ortomosaicos con un GSD máximo de 1.47 cm/píxel, lo que también limita la aplicación de esta investigación a ortomosaicos con un GSD dentro del rango de 0.54 a 1.59 cm/pixel. Esta selección reduce los vuelos históricos utilizables a 4 de los 29 disponibles, los cuales se resumen en la Tabla 4.

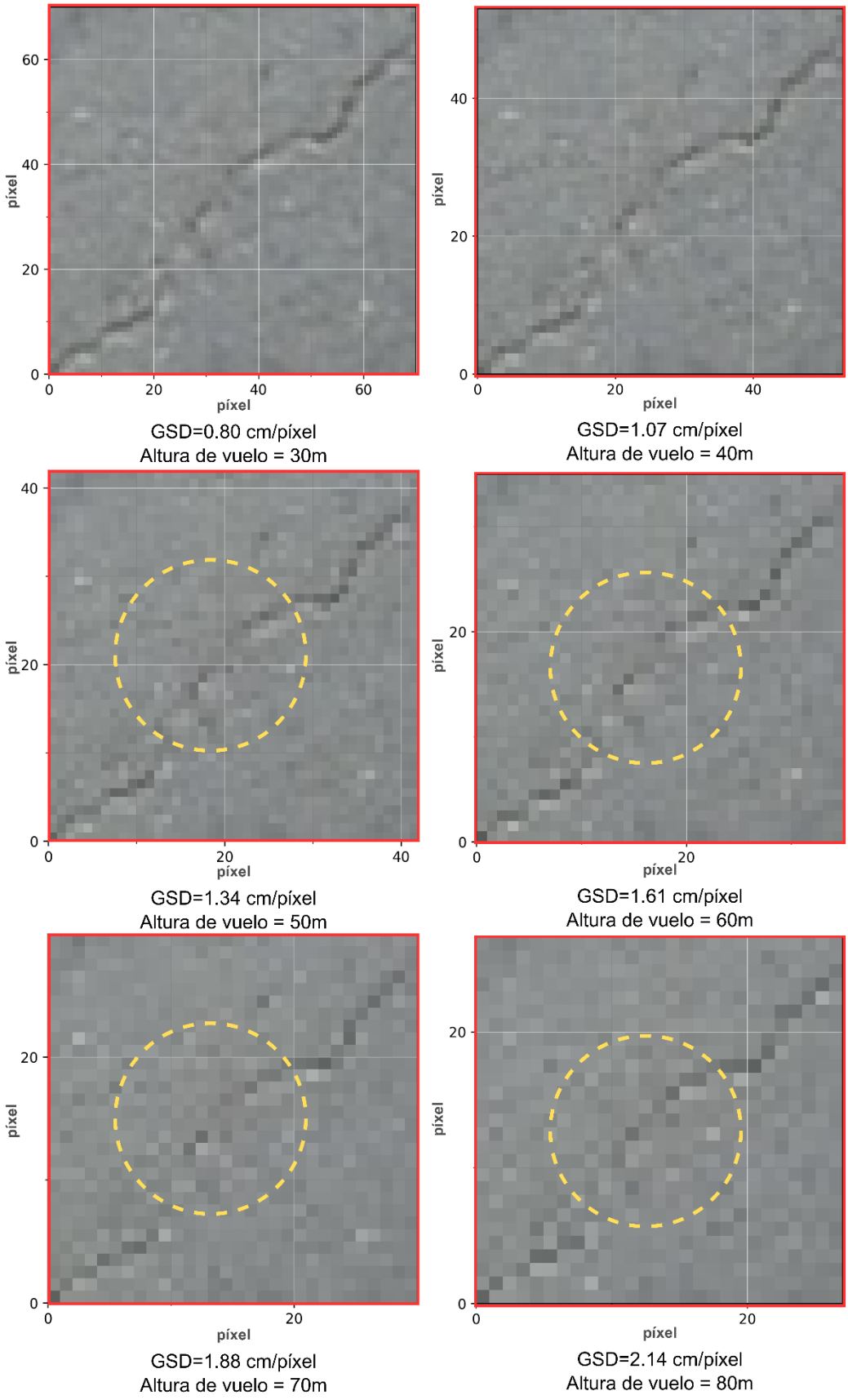


Figura 13: Resultado de la imagen simulando diferentes alturas de vuelo.

Finalmente, cada uno de ortomosaicos seleccionados se recortaron en imágenes de 448x448 pixeles, logrando así obtener una variedad de datos adecuados para el entrenamiento del modelo. Durante este este recorte, solo se consideraron las partes que contienen información, ya que, como se muestra en la Figura 14, las áreas del ortomosaico que no contienen datos están rellenadas con un color negro, el cual no sería útil como dato para el entrenamiento.

Es importante mencionar que algunas de las imágenes generadas por los recortes contienen parcialmente áreas de pavimento combinadas con zonas negras. Estas imágenes también fueron incluidas en el conjunto de datos de entrenamiento, dado que, es probable que el modelo reciba como entrada imágenes que incluyan parcialmente áreas negras dentro del recuadro de 448x448 píxeles, por lo que considerar estos casos asegura que el modelo pueda adaptarse a diferentes imágenes de entrada.

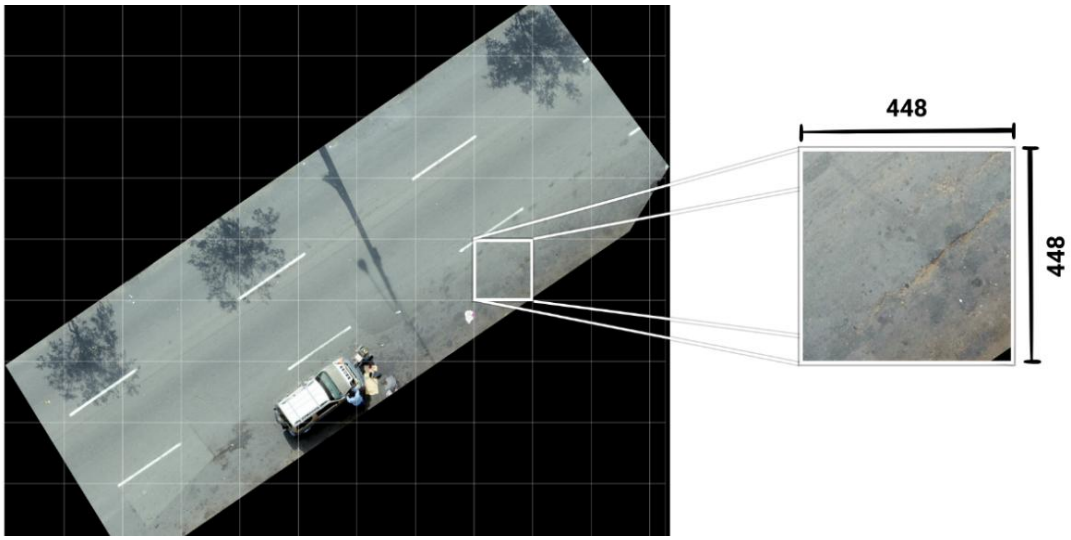


Figura 14: Ejemplo de recorte de ortomosaico 4972x4302 en imágenes de 448x448.

La cantidad de imágenes obtenidas a partir del recorte de ortomosaicos históricos seleccionadas, se detalla en la Tabla 4, en la columna titulada “Imágenes Obtenidas”, indicando un total de 8,006 imágenes.

3.2.1.2 Ajuste dimensional de las imágenes obtenidas con cámara

Antes de proceder con el recorte individual de las imágenes, se extrajo los fotogramas del video capturados a 60 fps, como se muestra en la Figura 15a. A continuación, se calculó el número adecuado de fotogramas para asegurar un cambio significativo entre cada imagen. Para ello, se consideraron como datos la velocidad del desplazamiento del vehículo, que era aproximadamente de 3 m/s (11 km/h) y el ancho captado por cada fotograma (A), estimado en 1 metro. Además, se estableció un traslape máximo del 25% entre imágenes (IT) para garantizar una diferencia adecuada entre ellas. Bajo este criterio, se formuló la siguiente ecuación, derivada de la Figura 15b.

$$(100\% - IT) * A * c + IT * A <= D$$

Donde:

A: Ancho captado por fotograma.

c: Cantidad de imágenes que se generaron en 1 segundo.

IT: Porcentaje de traslape por imagen.

D: Distancia recorrida del vehículo en 1 segundo.

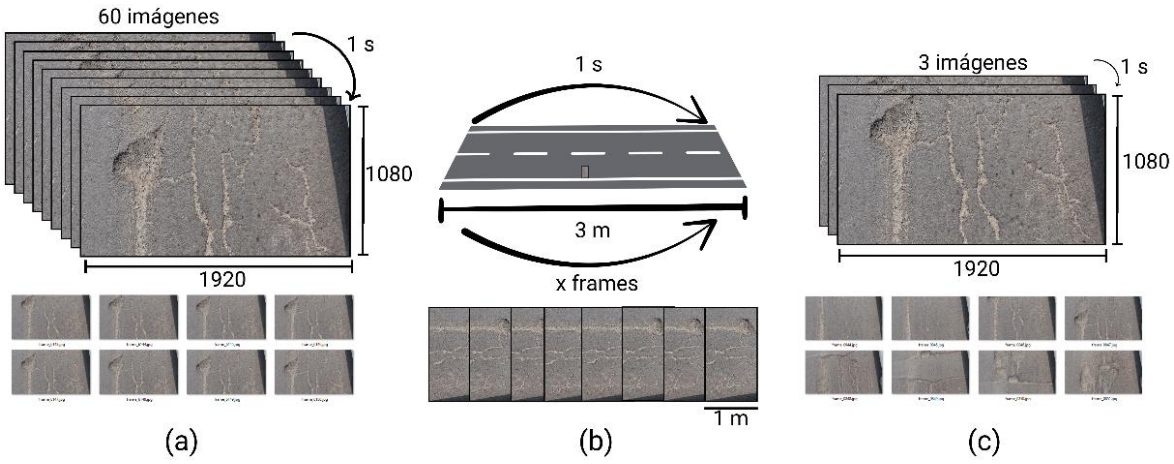


Figura 15: Proceso para la obtención de fotogramas con un cambio significativo entre imágenes.

Resolviendo la ecuación, se determinó que la cantidad adecuada de fotogramas por segundo para cumplir con un traslape menor o igual al 25% es de 3 fps, como se muestra en la Figura 15c.

Después de obtener los fotogramas, similares a la Figura 16a, cada uno de ellos fue recortado para generar 8 imágenes de 448x448 y 7 imágenes de menor tamaño, como se observa en la Figura 16b. Estas imágenes de menor tamaño también se utilizaron como datos de entrenamiento, ya que fueron completadas con pixeles negros, simulando recortes de ortomosaicos que contienen parcialmente pavimento en su superficie, como se ilustra en la Figura 16c.

En total se obtuvieron fueron 14,892 imágenes recortadas, como se detalla en la Tabla 5.

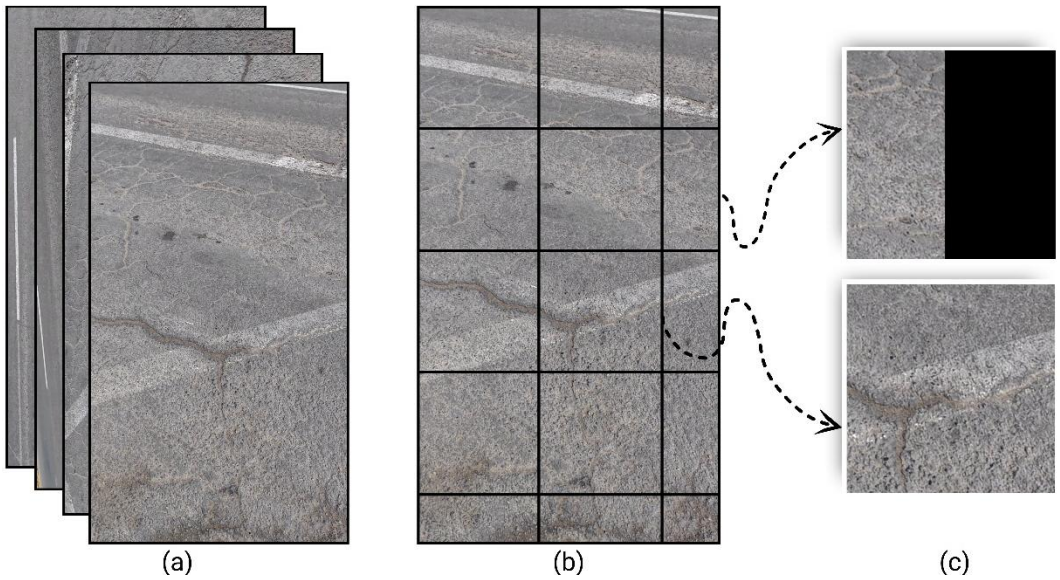


Figura 16: Recorte de Imágenes obtenidas mediante la cámara de alta resolución.

3.2.2 Categorización de imágenes de entrenamiento

Se revisaron cada una de las imágenes obtenidas a partir de los recortes, seleccionando aquellas que contenían fallas. Por defecto, las imágenes no seleccionadas fueron clasificadas como imágenes sin fallas. En esta categorización, no se incluyeron imágenes de cámara en la categoría imágenes sin fallas, pues la base de datos a partir de los ortomosaicos fue suficiente para mantener un equilibrio adecuado en el conjunto de datos.

Además, no se utilizó la totalidad de las imágenes sin fallas, ya que era necesario garantizar un equilibrio entre ambas categorías. Dado que las imágenes con fallas o grietas incluyen una proporción significativa de píxeles sin fallas, se decidió utilizar aproximadamente el 35% de la cantidad de imágenes con fallas como cantidad de

imágenes sin fallas, asegurando así un equilibrio adecuado en la base de datos. Durante la selección de las imágenes, se dio prioridad a aquellas que mostraban diferencias significativas, garantizando así la diversidad y representatividad de los datos. Tal y como se observa en la Figura 17, no solo se incluyeron imágenes del pavimento sin fallas, sino que también imágenes que correspondían a elementos muy cercanos al pavimento como veredas, camellones y marcas viales. Esto con la finalidad de representar mejor las condiciones variadas del entorno.

Figura 17: Ejemplos de imágenes sin grietas de la base de entrenamiento.

3.2.2.1 Categorización de imágenes obtenidas a partir del ortomosaico

Los ortomosaicos seleccionados y recortados, generaron 8,006 imágenes, las cuales fueron revisadas individualmente. Tras esta revisión, se identificaron 820 imágenes que presentan grietas y se seleccionaron 300 imágenes sin grietas, especificándose la cantidad seleccionada para cada ortomosaico histórico en la Tabla 4.

Tabla 4: Resumen de ortomosaicos seleccionados según el GSD y su clasificación.

Distrito Referencia	**GSD (cm/píxel)**	**Imágenes**		
		Obtenidas	**Con fallas**	**Sin fallas**
San Martín de Porres Av. Eduardo de Habich	0.52	4,039	636	204
Comas Av. Maestro Peruano 3	1.39	1,536	73	23
Comas Av. Maestro Peruano 2	1.40	930	47	15
Comas Av. Maestro Peruano 1	1.41	1,501	64	23
TOTAL		**8,006**	**820**	**300**

3.2.2.2 Categorización de imágenes obtenidas con cámara

Las imágenes recortadas obtenidas con cámara fueron 14,892 imágenes, de las cuales, tras la selección, quedaron 3,450 imágenes con grietas. Sin embargo, no se utilizaron todas estas imágenes, ya que, como se mencionó anteriormente, estas solo se utilizaron para complementar la base de datos, pues el objetivo principal de esta investigación es la detección de grietas en ortomosaicos, por lo que un exceso de imágenes diferentes a estas podría afectar negativamente en los resultados del modelo. Por lo tanto, solo se seleccionó 447 imágenes a partir de las 3,450 disponibles de manera aleatoria. En la Tabla 5 se presenta un resumen de lo señalado anteriormente.

Tabla 5: Resumen de Imágenes obtenidas mediante cámara y su clasificación

Distrito **Referencia**	**Recorte**	**Imágenes**		
		Con fallas	**Con fallas Seleccionadas**	**Sin fallas**
Ancón Av. Julio César Tello	14,892	3,450	447	0
TOTAL	**14,892**	**3,450**	**447**	**0**

3.3 Generación de máscaras de segmentación

La segmentación manual de máscaras se realizó para las imágenes seleccionadas, tanto para las imágenes obtenidas a partir de los ortomosaicos como para aquellas derivadas de los videos, utilizando el software Adobe Photoshop.

El procedimiento comenzó con la importación de las imágenes al software, donde se creó una capa separada para almacenar la segmentación, evitando modificaciones sobre la imagen base. Utilizando la herramienta pincel en color negro, se segmentaron manualmente las grietas, píxel por píxel, hasta delinear completamente las áreas agrietadas. Una vez finalizada la segmentación, se desactivó la visualización de la imagen base, lo que automáticamente dejó las áreas no segmentadas con un fondo blanco. Finalmente, se invirtieron los colores de la máscara, de modo que los píxeles blancos representaran las grietas y los píxeles negros indicaran las áreas sin grietas.

Este proceso se detalla en la Figura 18, donde la primera columna (a) presenta la imagen original, la segunda columna (b) muestra la delimitación de grietas, y la tercera columna (c) presenta la máscara final tras la inversión de colores.

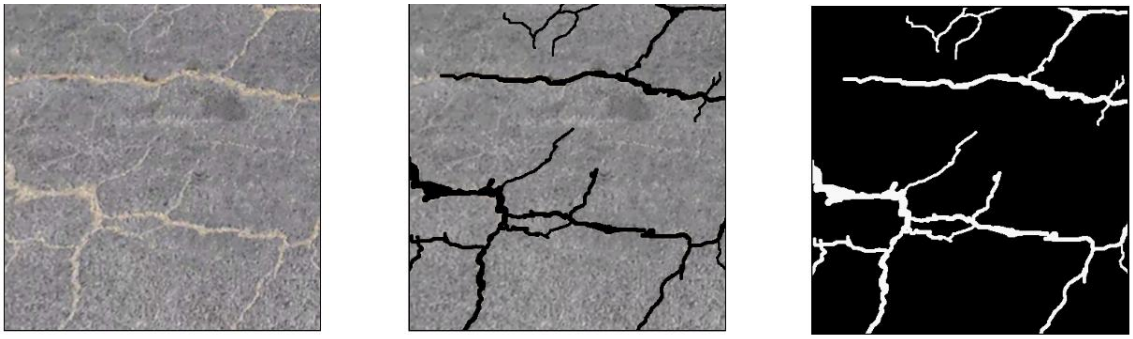


Figura 18: Generación de máscaras en las imágenes de entrenamiento.

Para las imágenes de menor tamaño, las máscaras se completaron con píxeles negro, hasta alcanzar el tamaño uniforme de 448x448 píxeles. Es importante destacar que la segmentación solo se aplicó a las imágenes previamente clasificadas como agrietadas, mientras que las imágenes sin grietas se mantuvieron con un fondo completamente negro, representando las áreas sin grietas en el conjunto de datos.

3.4 Consolidación y ampliación de datos para la segmentación de grietas

El procesamiento de imágenes descrito en las secciones anteriores resultó en una base de datos compuesta por 1,567 imágenes, de las cuales 447 imágenes se obtuvieron con cámara de alta resolución, las cuales todas presentaban grietas como se mencionó en las secciones anteriores, y 1,120 imágenes a partir de los ortomosaicos. Estas últimas se dividieron en 820 imágenes con grietas y 300 imágenes sin grietas.

Para mejorar la robustez del modelo y aumentar la diversidad del conjunto de datos, se aplicaron técnicas de aumento de datos (*data augmentation*). Las transformaciones consideradas incluyeron combinaciones de rotación (0°, 90°, 180 y 270°) junto con ajustes de brillo y contraste en ±0.2. Se excluyó la combinación de rotación de 0° sin cambios en el brillo o contraste, ya que generaría imágenes idénticas a las originales. Estas transformaciones se aplicaron aleatoriamente en cada imagen, generando 4 variaciones

por imagen según las combinaciones mencionadas, lo que resultó en un total de 6,268 imágenes generadas mediante el proceso de *data augmentation*.

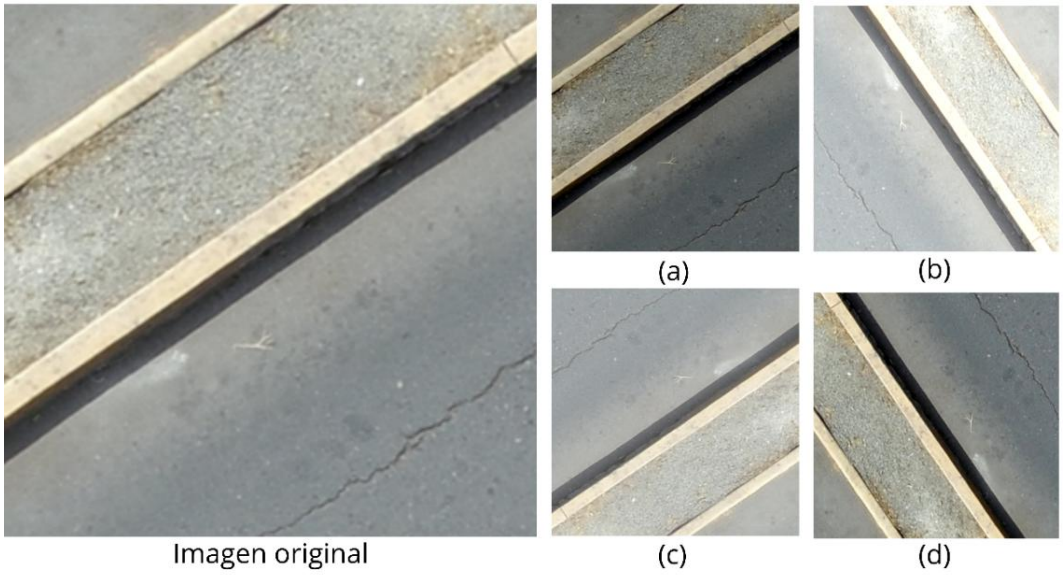


Figura 19: Ejemplo de transformaciones generadas mediante técnicas de aumento de datos.

En la Figura 19 se muestra la imagen original junto a 4 variaciones aleatorias de esta. En la imagen a no se observa una rotación, pero si una disminución en el brillo. La imagen b presenta una rotación de 90° acompañada de un aumento del brillo. Por otro lado, en la imagen c se observa una rotación de 180° combinada con una disminución en el contraste. Finalmente, en la imagen d muestra una rotación de 270° junto con un aumento del contraste.

En total, considerando tanto las imágenes generadas artificialmente como las imágenes originales, se generó un conjunto de 7,835 imágenes para la base de datos del modelo. Un resumen de esta información se presenta en la Tabla 6.

Tabla 6: Resumen de la base de datos: imágenes originales y generadas mediante técnicas de aumento de datos.

FUENTE	Imágenes con fallas	Imágenes sin fallas	TOTAL
Elaboración propia	1,267	300	1,567
Data augmentation	5,068	1,200	6,268
TOTAL	**6,335**	**1,500**	**7,835**

Finalmente, en la Figura 20 se presentan ejemplos representativos de las imágenes utilizadas junto con sus respectivas máscaras de segmentación, organizadas para mostrar las diferentes categorías de datos incluidas en el conjunto.

Las dos primeras filas contienen imágenes con grietas obtenidas a partir de recortes de ortomosaicos, acompañadas de sus respectivas máscaras de segmentación. En las filas posteriores, las dos primeras columnas muestran imágenes recortadas capturadas con cámara: la primera columna presenta una imagen completa, mientras que la segunda columna muestra una imagen parcialmente rellenada de manera artificial. Por último, las dos últimas columnas incluyen imágenes sin grietas junto a sus respectivas máscaras de segmentación, que muestran cómo se representan estas áreas en el modelo.

Figura 20: Ejemplo de imágenes de la base de datos con su respectiva máscara de segmentación.

Capítulo IV. Machine Learning para la detección de fallas en el pavimento

4.1 Arquitecturas seleccionadas para la comparativa y análisis de modelos

Entre las arquitecturas más utilizadas recientemente para tareas de segmentación, U-Net, Shallow U-Net, VGG16 U-Net, ResNet101 U-Net y DeepLabV3+ se destacan por su amplio reconocimiento y desempeño en diversas aplicaciones.

La arquitectura U-Net, ha sido reconocida como una de las arquitecturas más efectivas en tareas de segmentación. Sus creadores, Ronneberger et al. (2015) reportaron valores de *Intersection over Union* (IoU) del 92% en su aplicación inicial. Más recientemente Panella et al. (2022) destacó a U-Net como el estado del arte en la segmentación de grietas debido a su capacidad de alcanzar un alto rendimiento con tiempos de procesamiento relativamente cortos, superando a redes que no utilizan capas de *pooling*. Además, Liu et al. (2019) evaluaron U-Net en la segmentación de grietas en concreto armado utilizando un conjunto reducido de 57 imágenes, lograron un *F1-score* del 90%, incluso con datos limitados. Esto evidencia la capacidad de U-Net para alcanzar alta precisión en tareas de segmentación incluso cuando se dispone de datos limitados.

La Shallow U-Net es una versión menos profunda de U-Net que ofrece ventajas computacionales significativas. Sola & Scott (2022) demostraron que esta arquitectura reduce el uso de memoria en más del 50% en comparación con la U-Net completa, con una disminución mínima del 1% en el IoU. Además, la cantidad de parámetros es hasta diez veces menor, lo que la convierte en una solución eficiente para la segmentación de grietas en escenarios con recursos computacionales limitados.

Las arquitecturas basadas en modelos preentrenados han demostrado ser efectivas para mejorar la precisión en tareas de segmentación, todo ello gracias al aprovechar los pesos iniciales optimizados. Zhou et al. (2018) implementaron una variación de U-Net utilizando un modelo preentrenado, logrando un incremento promedio del 4% en el IoU respecto a U-

Net tradicional. En este contexto, ResNet 101-U-Net y VGG16 U-Net se destacan por su capacidad de integrar la arquitectura U-Net con modelos preentrenados en ImageNet, una base de datos que contiene más de 1.2 millones de imágenes clasificadas en 1,000 categorías. Este enfoque permite aprovechar el aprendizaje transferido, mejorando la eficiencia y el desempeño incluso con un conjunto de datos limitados.

Ghosh et al. (2021) demostraron que una variante de U-Net con VGG16 logró una precisión del 99.75% en la segmentación de tumores cerebrales, subrayando la eficacia del aprendizaje transferido en tareas de segmentación. Por otro lado, Putra & Suryanegara (2021) compararon la U-Net tradicional con Res U-Net, una versión mejorada de U-Net que incorpora bloques residuales similares a ResNet, logrando un incremento del 2% en la precisión. Esto sugiere que ResNet101 U-Net, al contar con una mayor profundidad y bloques residuales, podría ofrecer un rendimiento superior en la segmentación de grietas.

Sin embargo, aunque ResNet101 U-Net presenta una mayor cantidad de capas y profundidad, lo que puede conducir a una mayor precisión, también requiere un mayor costo computacional, mientras que VGG16 U-Net, al ser menos profunda y con una estructura más simple, podría ser más eficiente en términos de recursos computacionales, facilitando su implementación en entornos con limitaciones de hardware.

Finalmente, la arquitectura DeepLabV3+, también utiliza modelos preentrenados en ImageNet y ha demostrado un rendimiento notable en tareas de segmentación. Villarroel (2024) reportó que DeepLabV3+ superó a U-Net en la segmentación de cultivos de uva en imágenes satelitales, logrando una precisión global del 93.84% y un IoU de 0.8389. Estos resultados sugieren que DeepLabV3+ podría ofrecer un rendimiento favorable para la segmentación de grietas, especialmente al integrar el módulo *ASPP,* que permite identificar objetos de diversos tamaños y formas, como ocurre en las grietas.

En conclusión, la selección de estas cinco arquitecturas corresponde a su capacidad demostrada en tareas de segmentación, eficiencia en términos de recursos

computacionales y adaptabilidad a conjuntos de datos limitados. El análisis comparativo entre ellas permitirá identificar la opción óptima para la segmentación de grietas en el presente estudio. Para facilitar la comprensión de las diferencias entre ellas, se presenta un resumen gráfico a través de un diagrama de flujo en la Figura 21, donde se indica además la cantidad de parámetros para cada una de las arquitecturas.

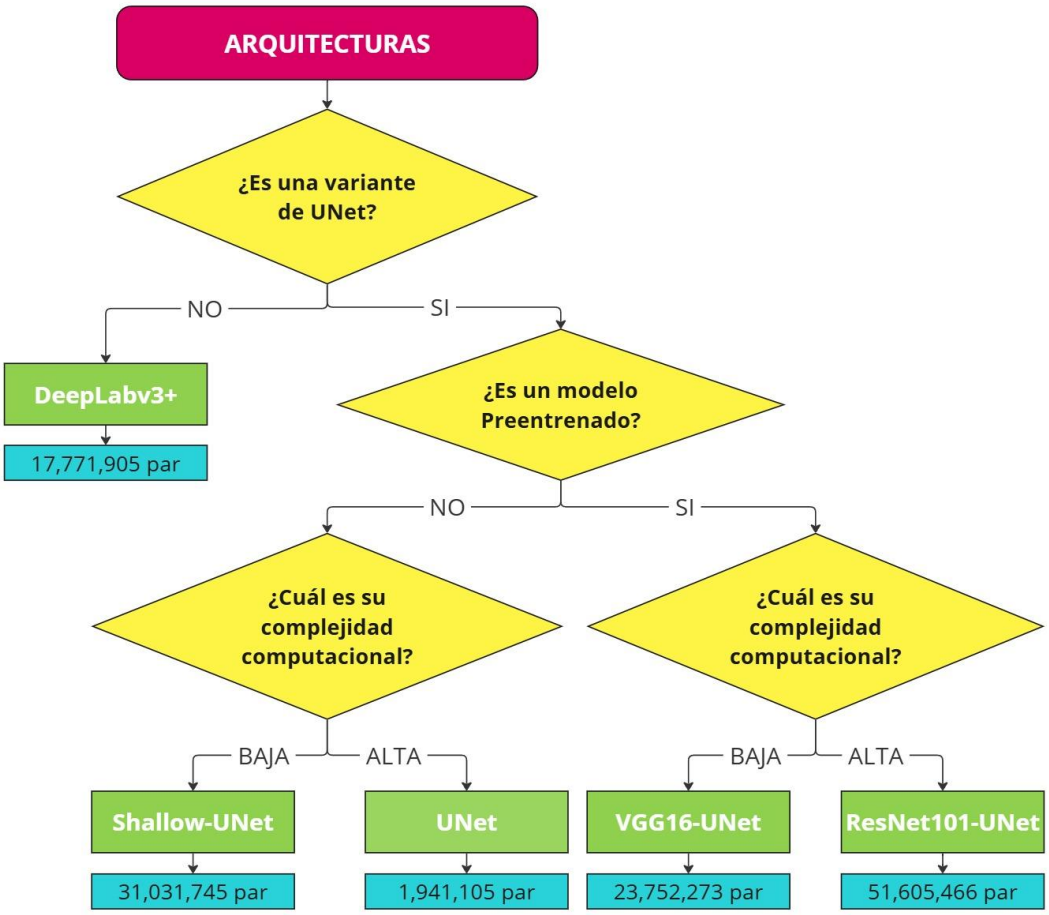


Figura 21: Diagrama de flujo de la selección de arquitecturas.

4.2 Selección de parámetros para el entrenamiento de los modelos

4.2.1 Criterios para la selección de hiperparámetros

En esta investigación, la selección de los hiperparámetros se realizó buscando un equilibrio entre la precisión del modelo y la eficiencia computacional. Los valores seleccionados fueron adaptados a las características del conjunto de datos y las limitaciones de hardware disponibles, las cuales se describen en las secciones posteriores.

4.2.1.1 Optimizador

Se utilizó el optimizador Adam debido a su capacidad para ajustar dinámicamente la tasa de aprendizaje, lo que permite una convergencia rápida y estable, incluso cuando la tasa de aprendizaje inicial no es óptima. Además, optimiza el uso de memoria en comparación con otros métodos, lo que lo hace adecuado para arquitecturas profundas (Diederik, 2014).

Estas características resultan particularmente beneficiosas en esta investigación, ya que el objetivo no es optimizar individualmente cada arquitectura mediante parámetros refinados, sino evaluarlas bajo condiciones estandarizadas, utilizando los mismos parámetros en todas ellas. Esto garantiza una comparación justa y objetiva del desempeño de cada modelo.

4.2.1.2 Tasa de aprendizaje

El optimizador Adam tiene una tasa de aprendizaje (*learning rate*) inicial por defecto de 0.0001 la cual es empleada comúnmente en investigaciones, logrando un equilibrio entre la convergencia rápida y la estabilidad de entrenamiento ya que se va adaptando cada lote de imágenes. (Diederik, 2014).

4.2.1.3 Número de épocas

Se estableció un máximo de 200 épocas (*epochs*) para el entrenamiento del modelo, un valor que demostró ser más que suficiente para los datos utilizados, ya que adicionalmente se implementó el *callback Early Stopping*, que detiene automáticamente el proceso de entrenamiento cuando la métrica de interés deja de mejorar durante un número determinado de épocas consecutivas. Los detalles sobre la configuración y uso de este *callback* se abordarán en las secciones posteriores.

4.2.1.4 Tamaño del lote

La configuración del tamaño del tamaño del lote (*batch size*) se estableció en 8 para todas las arquitecturas, buscando un equilibrio entre las cargas computacionales y la estabilidad

del entrenamiento. Esta decisión se tomó en función de las limitaciones de la capacidad computacional disponible, asegurando que el modelo pudiera entrenarse eficientemente sin comprometer los recursos disponibles.

4.2.1.5 Función de pérdida

La función de pérdida entropía cruzada binaria (*binary cross-entropy*) fue seleccionada para esta investigación debido a que funciona bien en tareas de segmentación binaria, donde cada píxel debe clasificarse como perteneciente o no a una grieta. Esta función mide la diferencia entre las probabilidades predichas por el modelo y las etiquetas reales, proporcionando una guía para ajustar los parámetros del modelo durante el entrenamiento.

Además, la eficacia de esta función de pérdida ha sido demostrada en investigaciones similares. Por ejemplo, Jing et al. (2024) utilizaron la entropía cruzada binaria para entrenar una variante mejorada de U-Net destinada a la detección de grietas en viviendas, logrando un IoU de hasta 91.7 %. Este resultado destaca la capacidad de esta función de pérdida para optimizar modelos en problemas relacionados con la segmentación binaria, como la detección precisa de grietas.

4.2.1.6 Función de activación

Según Goodfellow et al. (2016) en *Deep Learning*, la sigmoide es ampliamente utilizada en problemas de clasificación binaria debido a su capacidad para generar salidas probabilísticas. Esta característica hace que la sigmoide sea compatible con funciones de pérdida como la entropía cruzada binaria.

Por lo que en esta investigación se empleará la función de activación sigmoide en la capa de salida del modelo, ya que transforma las salidas del modelo en un rango entre 0 y 1, permitiendo interpretarlas como probabilidades de que un píxel pertenezca a la clase de “grieta”.

4.2.2 *Callbacks* implementados en el entrenamiento

Para esta investigación se seleccionaron dos tipos de *callbacks*, ajustados a las necesidades específicas del entrenamiento del modelo. A continuación, se describe cada uno de ellos.

4.2.2.1 Criterio de finalización temprana

Se identificó la necesidad de emplear el *callback* criterio de finalización temprana o *Early Stopping*, para optimizar el tiempo de entrenamiento, dado que se trabajó con múltiples modelos de *deep learning*. Este *callback* detiene automáticamente el entrenamiento cuando la métrica monitoreada, en este caso la pérdida en la validación deja de mejorar durante un número consecutivo de épocas definido por el parámetro *patience*.

La elección de la pérdida en los datos de validación (*val_loss*) como métrica monitoreada se justifica debido a la función de pérdida utilizada en esta investigación, *Binary Cross-Entropy*, ya que monitorear esta métrica asegura una correspondencia directa entre el objetivo del entrenamiento y la métrica monitoreada, garantizando coherencia en el proceso de optimización.

En esta investigación, se estableció un valor de 10 para el parámetro *patience*, permitiendo al modelo el tiempo necesario para estabilizar su aprendizaje sin extender innecesariamente el proceso de entrenamiento. Este valor proporciona un equilibrio razonable, ofreciendo suficientes oportunidades para que el modelo mejore mientras se evita el consumo excesivo de recursos computacionales

Adicionalmente, *Early Stopping* también permitió la restauración automática de los mejores pesos logrados durante el entrenamiento, asegurando que el modelo final entregara el mejor desempeño alcanzado.

4.2.2.2 Registro del proceso de entrenamiento

Se identificó la necesidad de emplear un registro del proceso de entrenamiento mediante el *callback CSVLogger*, ya que permite almacenar el historial de entrenamiento en un archivo CSV. Este registro facilita la generación posterior de gráficas que ofrecen una visualización clara del progreso del modelo durante el entrenamiento, incluyendo métricas como IoU, F1-score y *val_loss*, lo que a su vez permite realizar un análisis más profundo y fundamentado sobre el desempeño del modelo.

4.2.3 División del conjunto de datos

Según Goodfellow et al. (2016), asignar el 80% de los datos al entrenamiento es fundamental para evitar el sobreajuste y mejorar la capacidad de generalización del modelo.

Para esta investigación, los datos se dividieron en un 80 % para entrenamiento, 10 % para validación y 10 % para prueba, resultando en 6,268, 783 y 783 imágenes, respectivamente.

4.3 Implementación del proceso de entrenamiento del modelo

4.3.1 Características computacionales

El entrenamiento de los modelos se llevó a cabo en una computadora equipada con un procesador Intel®Xeon®Gold 6136 de 28 núcleos a una frecuencia de 3.00 GHz, 128 GB de memoria RAM DDR4 a 2666 MHz, y una tarjeta gráfica NVIDIA RTX A4000 con 16 GB de memoria VRAM. Los algoritmos de aprendizaje automático fueron implementados en el lenguaje de programación Python en su versión 3.11.2.

4.3.2 Implementación de modelos de entrenamiento

4.3.2.1 U-Net

El modelo U-Net fue implementado definiendo cada componente y operación de manera explícita, sin recurrir a herramientas de construcción de modelos. Este enfoque permitió un

control total sobre el diseño de la arquitectura y la configuración de sus parámetros. El proceso de implementación incluyó las capas convolucionales, operaciones de *pooling* y capas de *upsampling*, elementos característicos del diseño original propuesta por Ronneberger et al. (2015).

Durante la implementación, se configuraron manualmente los parámetros de cada capa, como el número de filtros asignados a cada bloque, que incrementaron progresivamente desde 64 en las primeras capas del *encoder* hasta 1024 en el *bottleneck*. Asimismo, los *kernels* convolucionales se definieron en 3x3 para las operaciones de convolución, y se utilizaron funciones de activación *ReLU* en cada capa para garantizar la no linealidad en el aprendizaje. Las operaciones de *pooling* y *upsampling* fueron configuradas con un tamaño de 2x2, asegurando una reducción o expansión uniforme en las dimensiones de los mapas de características a lo largo de la arquitectura.

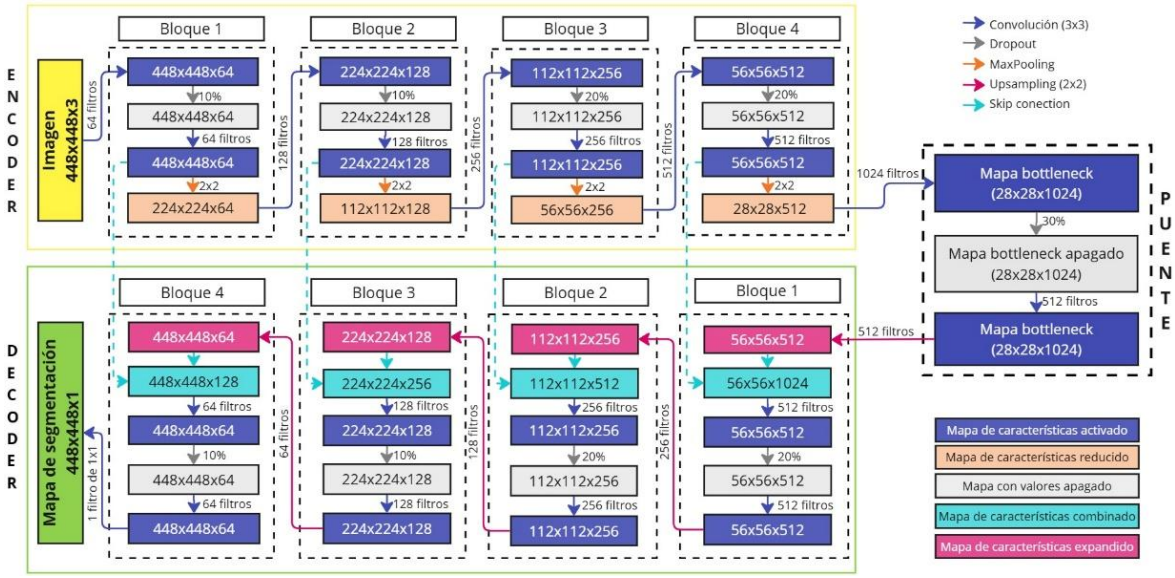


Figura 22: Diagrama del proceso de la arquitectura U-Net.

Para mitigar el riesgo de sobreajuste, se incluyeron capas de *dropout* con tasas ajustadas según la profundidad de la red. En las capas más superficiales se aplicó un *dropout* del 10 %, mientras que en las capas más profundas la tasa aumentó al 30 %. Además, las conexiones *skip* se implementaron para transferir directamente los mapas de características del *encoder* al *decoder*, lo que ayudó a preservar información espacial

importante durante el proceso de reconstrucción, lo que es una característica distintiva propia de U-Net.

En la Figura 22, se presenta una descripción gráfica del proceso del modelo, destacando la configuración de los parámetros clave y la estructura general de la arquitectura U-Net empleada en este trabajo, este diagrama ilustra cómo se organizan las diferentes capas, bloques y conexiones dentro del modelo.

4.3.2.2 Shallow U-Net

La arquitectura Shallow U-Net al igual que la U-Net, fue implementada de manera progresiva, definiendo cada componente de una manera individual. En la Figura 23, se presentan las configuraciones de esta arquitectura, donde se evidencia su profundidad reducida en comparación con la U-Net estándar.

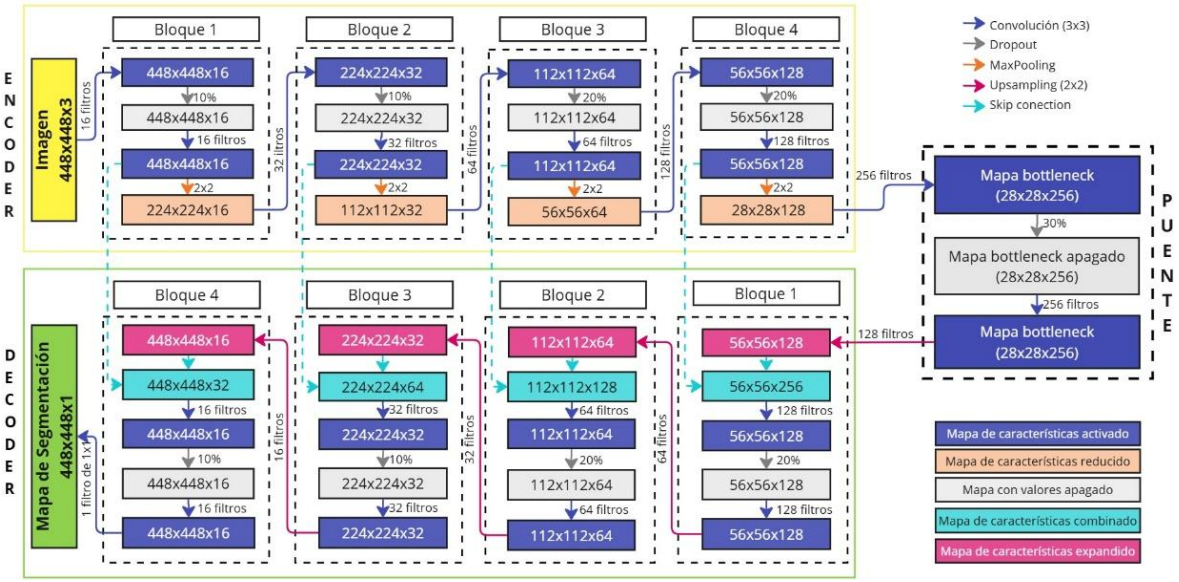


Figura 23: Diagrama del proceso de la arquitectura Shallow U-Net.

A diferencia de *U-Net*, *Shallow U-Net* se caracteriza por una reducción en el número de bloques en el enconder y el *decoder*, así como una menor cantidad de filtros en cada capa. En esta implementación, los bloques del *encoder* comienzan con 16 filtros y aumentan progresivamente hasta 256 en el *bottleneck*, mientras que en el *decoder* se realiza una reducción inversa, restaurando gradualmente las dimensiones originales del mapa de características. Además, las dimensiones de los mapas de características son más

compactas, como se observa en las capas convolucionales y de *pooling*. El uso de *dropout* también se ajusta, aplicando tasas de 10 % en las capas menos profundas y de 30 % en las capas más cercanas al *bottleneck*, para evitar el sobreajuste.

4.3.2.3 U-Net con backbones preentrenados

Entre las arquitecturas seleccionadas, tanto *VGG16 U-Net* como *ResNet101 U-Net* utilizan modelos preentrenados como *backbones*, los cuales potencian a la arquitectura base *U-Net*, para optimizar su desempeño en tareas de segmentación.

A diferencia de *U-Net* o *Shallow U-Net*, que fueron implementadas de manera progresiva, estas arquitecturas se implementaron utilizando la librería *segmentation_models* en *Python*. Esta herramienta ofrece una implementación optimizada que simplifica la integración de *backbones* con estructuras de segmentación como *U-Net*, entre ellos *VGG16* y *ResNet101*.

4.3.2.4 DeepLabV3+

La implementación de la arquitectura *DeepLabV3+* se llevó a cabo utilizando como *encoder* la red residual *ResNet-50*, que se encarga de extraer las características iniciales a partir de las imágenes. Estas características se envían al módulo *ASSP,* configurado con tasas de dilatación de 6, 12 y 18 para ampliar el campo receptivo de las convoluciones sin necesidad de reducir la resolución espacial. El módulo ASPP incluyó una operación de *pooling* global, que captura el contexto general de la imagen al extraer características globales.

Las salidas generadas por cada una de las convoluciones dilatadas, junto con la salida del *pooling* global, se concatenaron para combinar información multiescala del mapa de características, segmentando finamente las grietas y borde, así como patrones más grandes y globales en la imagen.

Para la etapa del decodificador, se mantuvo la configuración original propuesta en *DeepLabV3+,* sin realizar modificaciones adicionales. Esta etapa se encargó de refinar las características combinadas, reconstruyendo el mapa de segmentación final con una resolución que preserva los detalles espaciales relevantes para la tarea de segmentación.

4.3.3 Desarrollo del modelo en python para segmentación de grietas

El procedimiento para el entrenamiento de los modelos de segmentación de grietas con *Deep Learning* se desarrolló en varias etapas, siguiendo una secuencia estructurada. Estas etapas, detalladas en la Figura 24, incluyen desde la configuración inicial de los recursos computacionales y el *framework*, hasta la ejecución del entrenamiento propiamente dicho.

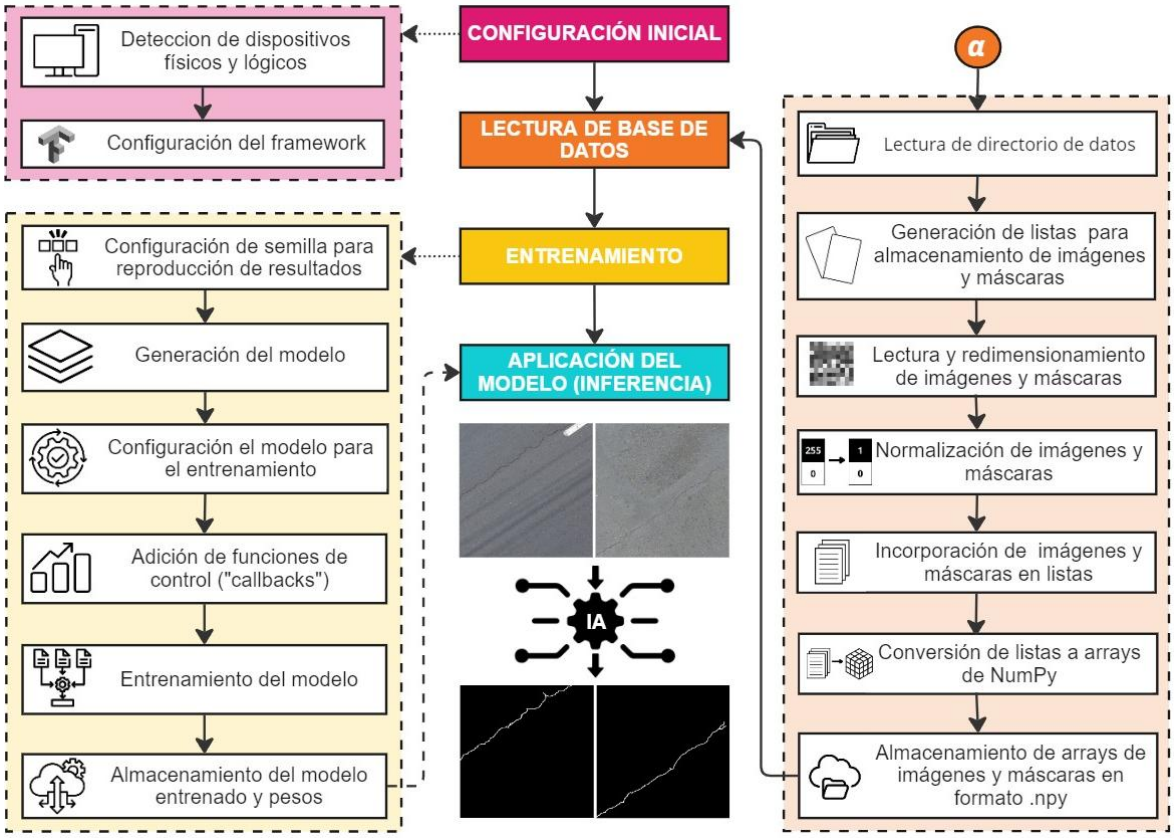


Figura 24: Diagrama del procedimiento estándar para el entrenamiento de los modelos.

4.3.3.1 Configuración inicial

La configuración inicial comenzó con la detección de dispositivos físicos y lógicos disponibles en el sistema, lo que permitió verificar que los recursos, como la Unidad del

procesamiento gráfico (GPU), estuvieran correctamente configurados y disponibles para optimizar los tiempos de entrenamiento.

Posteriormente, se llevó a cabo la configuración del *framework*, seleccionando la versión específica de *TensorFlow* compatible con los requerimientos. Además, se incluyeron librerías complementarias que trabajan en conjunto con *TensorFlow* para facilitar la implementación de arquitecturas.

4.3.3.2 Lectura de base de datos

Dado que el proceso de la lectura de imágenes se realizaría de manera iterativa en cada arquitectura, cada vez que el modelo se entrenaba, se decidió optimizar este procedimiento mediante la creación de un archivo en formato *Numpy*. Esto permitió simplificar el flujo de trabajo y reducir tiempos asociados a la lectura de imágenes.

Para generar este archivo, primero se realizó la lectura de las carpetas que contenían las imágenes de entrada y sus respectivas máscaras segmentadas. Se recolectaron los nombres de las imágenes y máscaras en listas para verificar que existiera correspondencia entre ambas. Posteriormente se redimensionaron las imágenes como las máscaras y se les aplicaron un proceso de normalización, ajustando los valores de los píxeles a un rango entre 0 y 1. Finalmente, las listas generadas fueron convertidas en arreglos de *NumPy*, lo que permitió almacenar los datos procesados de manera eficiente y reutilizarlos en futuros entrenamientos, evitando la necesidad de rehacer el preprocesamiento y garantizando consistencia en los datos utilizados.

4.3.3.3 Entrenamiento

El proceso de entrenamiento comenzó con la configuración de la semilla (*seed*) asegurando la reproducibilidad de los resultados en todas las arquitecturas evaluadas. Posteriormente, se realizó la generación del modelo mediante la definición de la arquitectura.

A continuación, se configuraron los *hiperparámetros* junto con las funciones de control (*callbacks*), que permitieron monitorear el progreso de entrenamiento, con todos los elementos configurados, se procedió al entrenamiento del modelo. Finalmente, una vez concluido el entrenamiento, se almacenaron tanto los pesos, el modelo entrenado y las métricas de entrenamiento, para poder emplearlos posteriormente.

4.3.3.4 Aplicación del modelo

El modelo entrenado se utilizó en la fase inferencia, donde se realizaron las predicciones sobre las imágenes del conjunto de testeo. Estas predicciones se almacenaron en una carpeta para su posterior análisis y comparación entre las diferentes arquitecturas.

4.3.4 Resultados del proceso de entrenamiento

Para una adecuada presentación de los resultados, se muestran las curvas de F1-Score, IoU y pérdidas correspondientes a las fases de entrenamiento y validación, evaluadas a lo largo de las épocas para cada una de las arquitecturas consideradas en la presente investigación.

4.3.4.1 Shallow U-Net

En la Figura 25 se ilustra el proceso de entrenamiento del modelo, en el cual el callback *early stopping* detuvo automáticamente el entrenamiento en la época 59, tras detectar que la pérdida en el conjunto de imágenes de validación dejó de mejorar consistentemente a partir de la época 49, donde se alcanzaron los mejores pesos del entrenamiento. En este punto el modelo logró un F1-Score de 0.7618 y un IoU de 0.6188, mientras que la pérdida en validación alcanzó un valor mínimo de 0.0287.

Durante las 10 primeras épocas, se observó un crecimiento constante en las curvas IoU y F1-Score, así como una disminución rápida en el valor de la pérdida, indicando que el modelo aprendió de manera eficiente en las etapas iniciales del entrenamiento. A partir de la época 20, aunque las curvas de IoU y F1-Score continuaron creciendo, lo hicieron a un

ritmo más lento, sugiriendo una reducción en la ganancia de aprendizaje. De manera similar las curvas de pérdida comenzaron a estabilizarse.

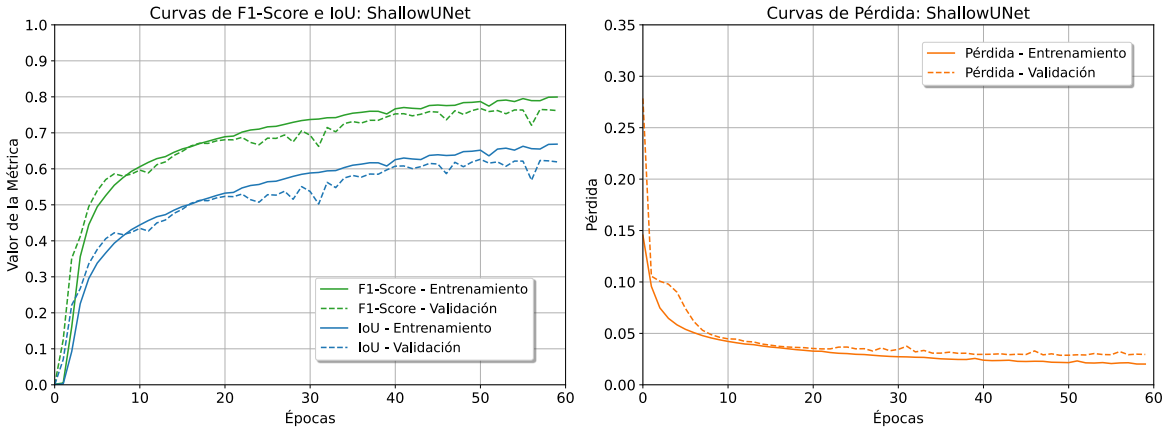


Figura 25: Métricas de entrenamiento por época de la arquitectura Shallow U-Net.

4.3.4.2 U-Net

En la Figura 26 se ilustra el proceso de entrenamiento del modelo, en el cual el *callback early stopping* detuvo automáticamente el entrenamiento en la época 34, tras detectar que la pérdida en el conjunto de imágenes de validación dejó de mejorar consistentemente a partir de la época 24, donde se alcanzaron los mejores pesos del entrenamiento. En este punto el modelo logró un F1-Score de 0.7787 y un IoU de 0.6417, mientras que la pérdida en validación alcanzó un valor mínimo de 0.0267.

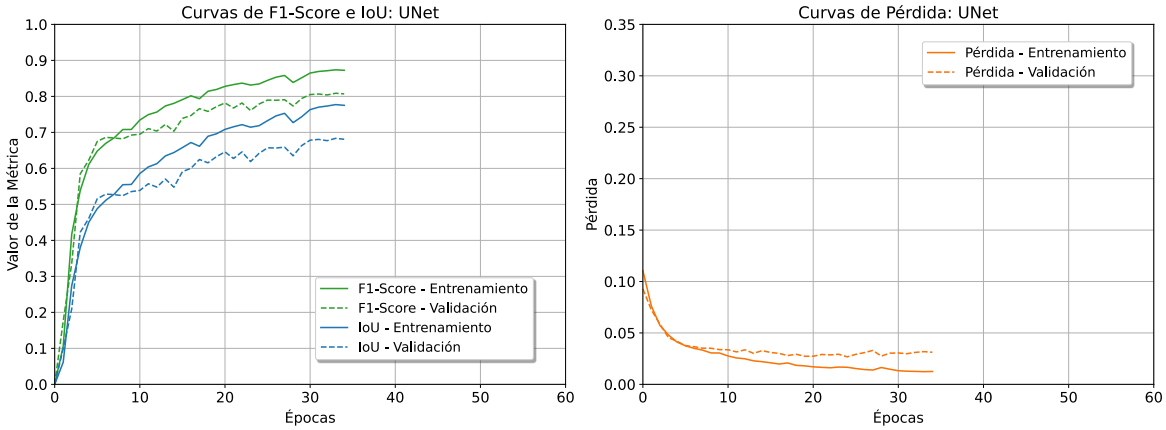


Figura 26: Métricas de entrenamiento por época de la arquitectura U-Net.

Durante las 7 primeras épocas, se observó un crecimiento constante en las curvas IoU y F1-Score, así como una disminución rápida en el valor de la pérdida, indicando que el modelo aprendió de manera eficiente en las etapas iniciales del entrenamiento. A partir de

la época 10, aunque las curvas de IoU y F1-Score continuaron creciendo, lo hicieron a un ritmo más lento, sugiriendo una reducción en la ganancia de aprendizaje. De manera similar las curvas de pérdida comenzaron a estabilizarse.

4.3.4.3 VGG16 U-Net

En la Figura 27 se ilustra el proceso de entrenamiento del modelo, en el cual el *callback early stopping* detuvo automáticamente el entrenamiento en la época 24, tras detectar que la pérdida en el conjunto de imágenes de validación dejó de mejorar consistentemente a partir de la época 14, donde se alcanzaron los mejores pesos del entrenamiento. En este punto el modelo logró un F1-Score de 0.7983 y un IoU de 0.6678, mientras que la pérdida en validación alcanzó un valor mínimo de 0.0243.

Durante las 3 primeras épocas, se observó un crecimiento constante en las curvas IoU y F1-Score, así como una disminución rápida en el valor de la pérdida, indicando que el modelo aprendió de manera eficiente en las etapas iniciales del entrenamiento. A partir de la época 8, aunque las curvas de IoU y F1-Score continuaron creciendo, lo hicieron a un ritmo más lento, sugiriendo una reducción en la ganancia de aprendizaje. De manera similar las curvas de pérdida comenzaron a estabilizarse.

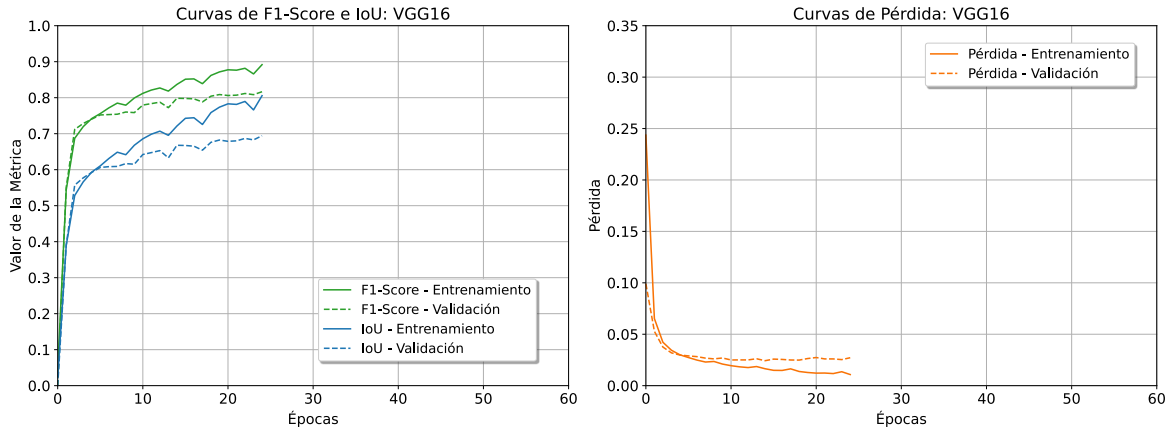


Figura 27: Métricas de entrenamiento por época de la arquitectura VGG16 U-Net.

4.3.4.4 ResNet101 U-Net

En la Figura 28 se ilustra el proceso de entrenamiento del modelo, en el cual el *callback early stopping* detuvo automáticamente el entrenamiento en la época 30, tras detectar que

la pérdida en el conjunto de imágenes de validación dejó de mejorar consistentemente a partir de la época 20, donde se alcanzaron los mejores pesos del entrenamiento. En este punto el modelo logró un F1-Score de 0.8161 y un IoU de 0.693, mientras que la pérdida en validación alcanzó un valor mínimo de 0.0234.

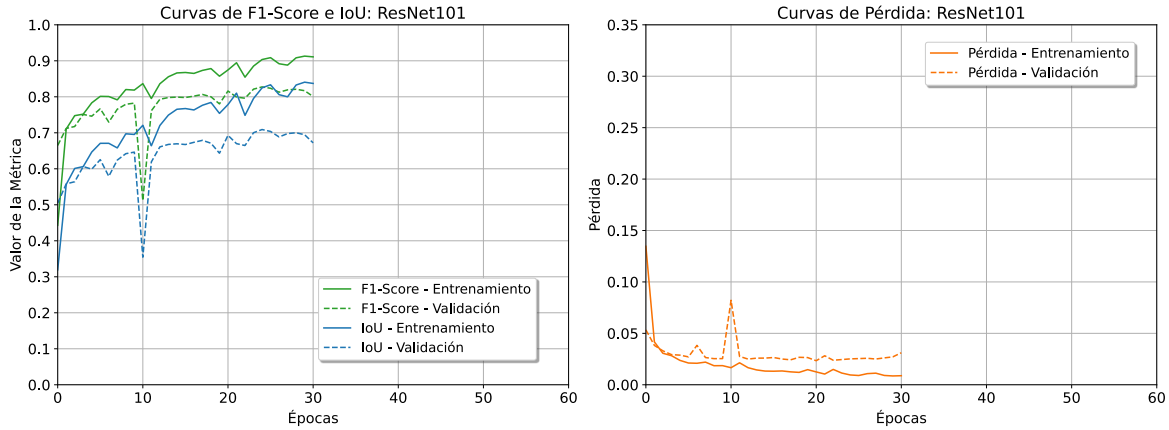


Figura 28: Métricas de entrenamiento por época de la arquitectura ResNet101 U-Net.

Durante las 2 primeras épocas, se observó un crecimiento constante en las curvas IoU y F1-Score, así como una disminución rápida en el valor de la pérdida, indicando que el modelo aprendió de manera eficiente en las etapas iniciales del entrenamiento. A partir de la época 11, aunque las curvas de IoU y F1-Score continuaron creciendo, lo hicieron a un ritmo más lento, sugiriendo una reducción en la ganancia de aprendizaje. De manera similar las curvas de pérdida comenzaron a estabilizarse.

4.3.4.5 DeepLabv3+

En la Figura 29 se ilustra el proceso de entrenamiento del modelo, en el cual el *callback early stopping* detuvo automáticamente el entrenamiento en la época 16, tras detectar que la pérdida en el conjunto de imágenes de validación dejó de mejorar consistentemente a partir de la época 6, donde se alcanzaron los mejores pesos del entrenamiento. En este punto el modelo logró un F1-Score de 0.7874 y un IoU de 0.6529, mientras que la pérdida en validación alcanzó un valor mínimo de 0.0239.

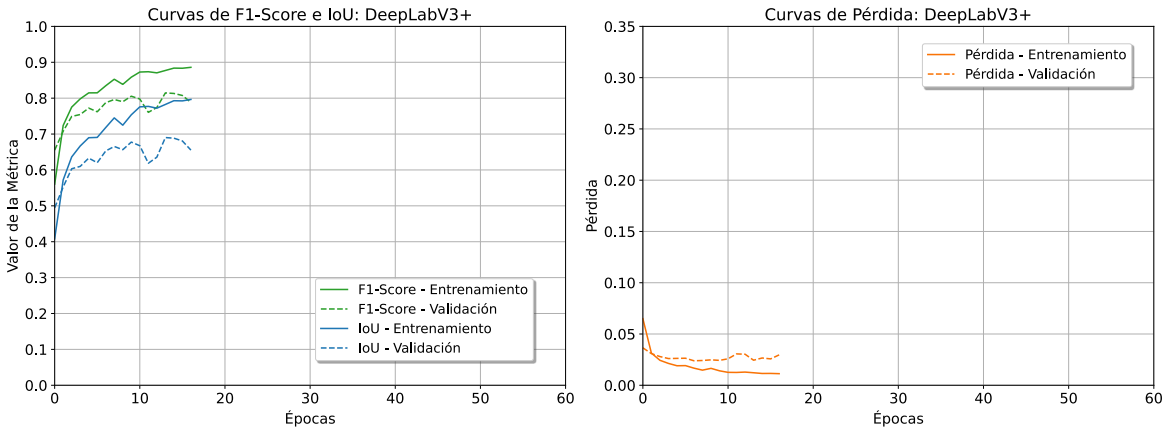


Figura 29: Métricas de entrenamiento por época de la arquitectura DeepLabv3+.

Durante las 2 primeras épocas, se observó un crecimiento constante en las curvas IoU y F1-Score, así como una disminución rápida en el valor de la pérdida, indicando que el modelo aprendió de manera eficiente en las etapas iniciales del entrenamiento. A partir de la época 2, aunque las curvas de IoU y F1-Score continuaron creciendo, lo hicieron a un ritmo más lento, sugiriendo una reducción en la ganancia de aprendizaje. De manera similar las curvas de pérdida comenzaron a estabilizarse.

4.3.5 Resumen de resultados del entrenamiento

A continuación, en la Tabla 7 se presenta un resumen comparativo para cada una de las arquitecturas evaluadas. Esta tabla incluye los valores mínimos de la métrica de pérdida alcanzados durante el entrenamiento y los valores máximos para las métricas F1-Score e Iou. Adicionalmente, se detalla el tiempo total de entrenamiento (TE) en minutos requerido por cada arquitectura y, en la última columna se reporta el tiempo promedio de inferencia (TI) en milisegundos, indicando cuanto tarda un modelo en predecir una imagen.

Tabla 7: Resumen de los resultados de entrenamiento.

Arquitectura	**Pérdida**	**Métricas de validación**		**TE**	**TI**
		F1-Score	**IoU**	**(minutos)**	**(ms/imagen)**
Shallow U-Net	0.0287	0.762	0.619	109.5	29.3
U-Net	0.0267	0.779	0.642	323.4	32.9
VGG16 U-Net	0.0243	0.798	0.668	142.65	28.4
ResNet101 U-Net	0.0234	0.816	0.693	177.6	37.8
DeepLabV3+	0.0239	0.787	0.653	72.5	30.5

4.4 Postprocesamiento de inferencia

El postprocesamiento de las inferencias consistió en binarizar las máscaras de salida, dado que estas representan probabilidades por píxel de pertenecer a la clase "grieta". La aplicación de un umbral permite convertir estas probabilidades en máscaras binarias, facilitando análisis posteriores y aplicaciones prácticas donde se requieran segmentaciones precisas. A partir del umbral seleccionado, los valores menores o iguales a este se asignan a 0 (representados en negro), mientras que los valores superiores se asignan a 1 (representados en blanco), generando así una máscara binaria que facilita la segmentación de grietas.

Inicialmente, se consideró utilizar un umbral fijo de 0.5, bajo la premisa de que los píxeles con valores superiores tendrían una mayor probabilidad de representar grietas. Aunque esta estrategia produjo resultados generalmente buenos, presentaba la limitación de omitir pequeñas secciones de las grietas, lo que generaba discontinuidades en la segmentación.

Para mejorar la adaptabilidad del umbral, se optó por un enfoque dinámico en el que cada inferencia tuviera su propio umbral. Para ello, se utilizó el percentil 90, priorizando las regiones con alta probabilidad de grietas y reduciendo la influencia de aquellas con valores menores. Sin embargo, este método no permitía una correcta adaptación del umbral en segmentaciones donde la máscara de probabilidades tenía valores bajos, lo que resultaba en una segmentación con un alto nivel de ruido. Ante esta situación, se consideró necesario realizar un análisis más detallado para optimizar la selección del umbral.

Para identificar el umbral adecuado, fue necesario considerar la distribución de valores en la máscara de probabilidades. En este proceso, se presentan dos escenarios distintos: en el primer caso, la imagen de salida no contiene grietas, lo que genera una distribución unimodal donde la mayoría de los valores están cercanos a 0. En el segundo caso, la imagen de salida sí contiene grietas, pero estas nunca ocupan toda la imagen, lo que da lugar a una distribución bimodal, donde un grupo de valores corresponde al fondo, donde

se tiene probabilidades bajas y otro a las grietas, donde se obtienen probabilidades más altas.

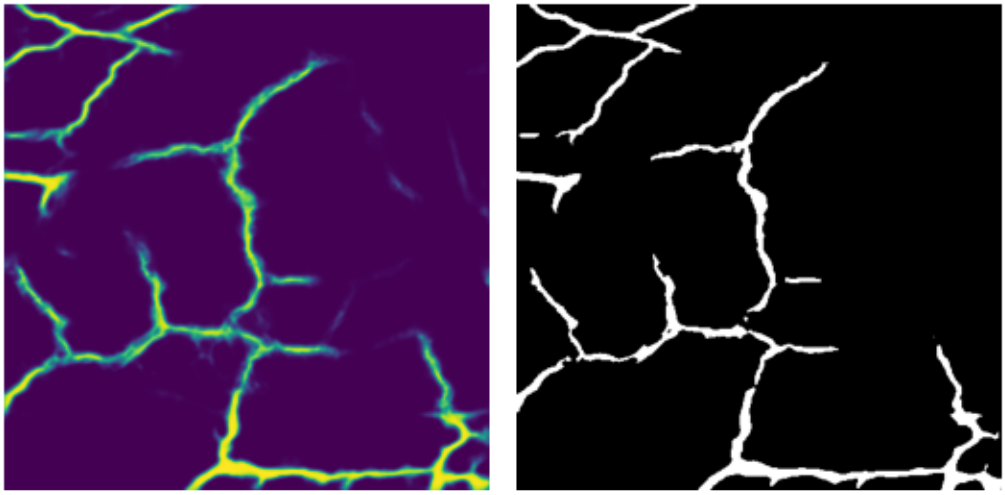


Figura 30: Máscaras se segmentación antes y después del postprocesamiento.

Dado que el umbral debe separar de manera precisa estos dos grupos, se empleó el método de Otsu como una solución óptima, ya que determina automáticamente el umbral que minimiza la varianza intra-clase, distinguiendo entre fondo y grietas, véase Figura 30. Sin embargo, en imágenes donde la distribución de probabilidades es unimodal, el umbral de Otsu tiende a ser demasiado bajo, lo que podría inducir a la segmentación errónea de ruido. Para evitar este problema, se estableció un umbral mínimo del 0.01, esta estrategia garantiza que el proceso de binarización mantenga su automatización, detectando correctamente las regiones de interés adecuadamente.

4.5 Comparación y análisis de resultados de los modelos entrenados

4.5.1 Comparación y análisis de las gráficas con las métricas de entrenamiento

En las gráficas de las métricas presentadas en las secciones anteriores, las variaciones en las épocas de finalización del entrenamiento representan las diferencias en la complejidad y el diseño de las arquitecturas, así como el número de parámetros de cada modelo. Mientras que Shallow U-Net requirió más iteraciones (época 59) para alcanzar un desempeño óptimo debido a su simplicidad estructural, arquitecturas más avanzadas como VGG16 U-Net (época 24), ResNet101 U-Net (época 30) y DeepLabV3+ (época 16)

lograron converger en menos épocas gracias a características como el uso de pesos preentrenados y diseños optimizados. Cabe destacar que la profundidad de las capas también influye, ya que modelos más profundos, como ResNet101 U-Net, necesitaron más iteraciones para ajustar sus parámetros, mientras que DeepLabV3+ se estabilizó rápidamente gracias a su diseño avanzado, que incluye convoluciones dilatadas y el módulo ASPP que captura características multiescala.

En cuanto a las métricas iniciales, como el IoU y el F1-Score, se observan diferencias notables entre las arquitecturas. Modelos como DeepLabV3+ y ResNet101 U-Net destacan por su eficiencia desde las primeras épocas, logrando métricas superiores debido a sus diseños avanzados y al uso de aprendizaje transferido, lo que facilita una rápida adaptación a los datos. Por otro lado, U-Net y Shallow U-Net presentan un inicio más lento, evidenciando su necesidad de más iteraciones para ajustarse completamente. VGG16 U-Net se posiciona en un punto intermedio, mostrando un desempeño inicial moderado pero consistente. Estos resultados subrayan cómo los modelos con backbones preentrenados pueden optimizar tanto el rendimiento inicial como el proceso general de aprendizaje.

Asimismo, las gráficas de F1-Score e IoU evidencian un patrón común en la mayoría de las arquitecturas, donde la curva de entrenamiento se sitúa por encima de la de validación, indicando una mejora progresiva en los datos de entrenamiento. Sin embargo, en Shallow U-Net y U-Net, durante las primeras 9 épocas, la curva de validación supera a la de entrenamiento, posiblemente debido a la limitada capacidad inicial de estas arquitecturas para capturar las complejidades de los datos. En modelos con backbones preentrenados, como ResNet101 U-Net y DeepLabV3+, este fenómeno es breve, restringiéndose a las primeras épocas.

Finalmente, se observó en las gráficas de la arquitectura ResNet101 U-Net, que en la época 10 la validación experimentó un incremento abrupto, posiblemente causado por un lote de datos atípicos. A pesar de este pico, el modelo logra estabilizarse a partir de la época 11 en adelante.

4.5.2 Comparativa del desempeño de las arquitecturas

4.5.2.1 Comparativa del desempeño en tiempos de entrenamiento y velocidad de inferencia

En la Figura 31 se tiene un resumen de los tiempos de entrenamiento, en donde se observa que, DeepLabV3+ se destaca como la arquitectura más eficiente, con solo 72.5 minutos, gracias a su diseño y su rápida convergencia en la época 16. Por el contrario, U-Net registró el mayor tiempo de entrenamiento, alcanzando 323.4 minutos, lo cual se le atribuye a su mayor cantidad de parámetros y su convergencia en la época 34. Shallow U-Net aunque completó su entrenamiento en 49 épocas, mostró un tiempo significativamente menor, con 109.5 minutos, reflejando su simplicidad estructural que permite un procesamiento más rápido por época.

Por otro lado, VGG16 U-Net y ResNet101 U-Net presentaron tiempos intermedios, de 142.7 y 177.6 minutos respectivamente. Esto se debe al uso de backbones preentrenados, que les permitió detenerse en menos épocas (24 y 30 respectivamente), aunque sus mayores profundidades aumentaron los requerimientos computacionales.

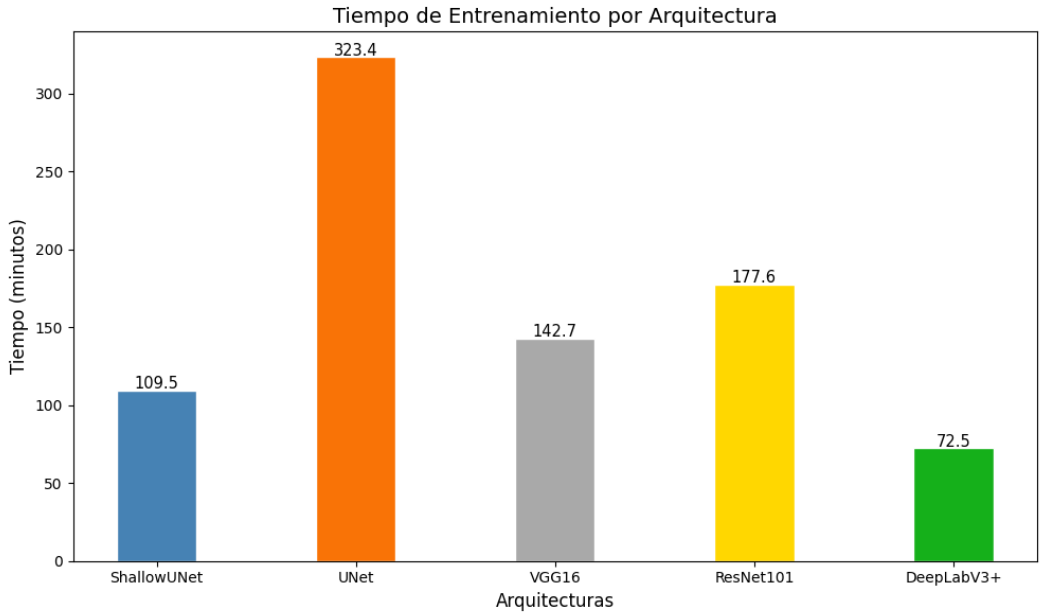


Figura 31: Gráfica de tiempos de entrenamiento en minutos de cada arquitectura.

En la Figura 32, que muestra la velocidad de inferencia, DeepLabV3+ se posiciona como la arquitectura más rápida, procesando aproximadamente 32 imágenes por segundo, lo que resalta su eficiencia computacional durante la fase de inferencia. Le sigue Shallow U-Net con una velocidad de cerca de 30 imágenes por segundo, sustentada por su simplicidad estructural que reduce los tiempos de procesamiento. En un rango intermedio se encuentra U-Net, con una velocidad de inferencia ligeramente menor, alrededor de 26 imágenes por segundo, debido a su mayor número de parámetros en comparación con Shallow U-Net. VGG16 U-Net y ResNet101 U-Net presentaron velocidades inferiores, con aproximadamente 22 y 20 imágenes por segundo, respectivamente, debido a la mayor complejidad y profundidad de sus backbones preentrenados. Estos resultados reflejan cómo la complejidad de cada arquitectura influye directamente en su desempeño durante la inferencia, siendo más rápidas las arquitecturas menos profundas o aquellas diseñadas específicamente para optimizar el procesamiento computacional.

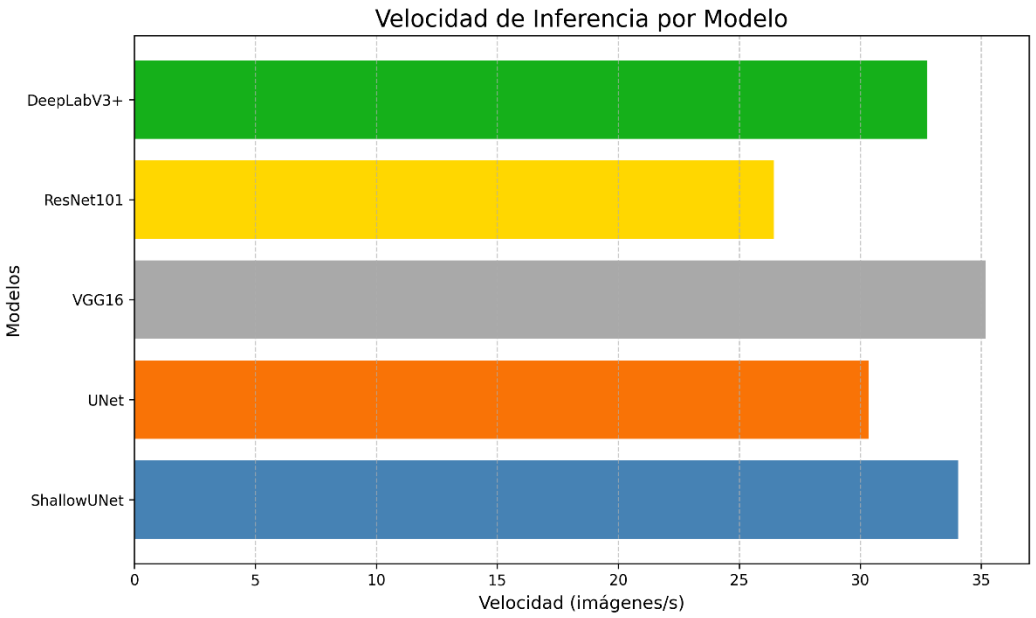


Figura 32: Gráfica de velocidades de inferencia por imágenes por segundo de cada arquitectura.

De los resultados del tiempo de entrenamiento y velocidad de inferencia se deduce que, las arquitecturas más simples o diseñadas específicamente para optimizar el procesamiento (Shallow U-Net, DeepLabV3+) muestran una correlación entre su menor tiempo de entrenamiento y mayor velocidad de inferencia. Por otro lado, arquitecturas más

complejas y con mayores parámetros, como ResNet101 U-Net, tienden a ser más lentas en ambas etapas debido a su carga computacional.

4.5.2.2 Comparativa de desempeño basada en métricas de entrenamiento

La Figura 33 muestra el desempeño de cada modelo en la validación de las métricas F1-Score e IoU. ResNet101 U-Net destacó alcanzando un valor de F1-Score de 0.816 e IoU de 0.693, destacando su capacidad para capturar detalles relevantes debido a su profundidad y al uso de un backbone preentrenado. VGG16 U-Net también obtuvo resultados competitivos, con un F1-Score de 0.798 e IoU de 0.668, demostrando que, aunque tiene menor profundidad en comparación con ResNet101 U-Net, sigue siendo eficiente. En contraste, U-Net y Shallow U-Net, obtuvieron valores más bajos, con un F1-Score de 0.779 y 0.762, e IoU de 0.642 y 0.619, respectivamente. Evidenciando que, aunque estas arquitecturas son efectivas, la ausencia de pesos entrenados limita su rendimiento frente a arquitecturas más avanzadas.

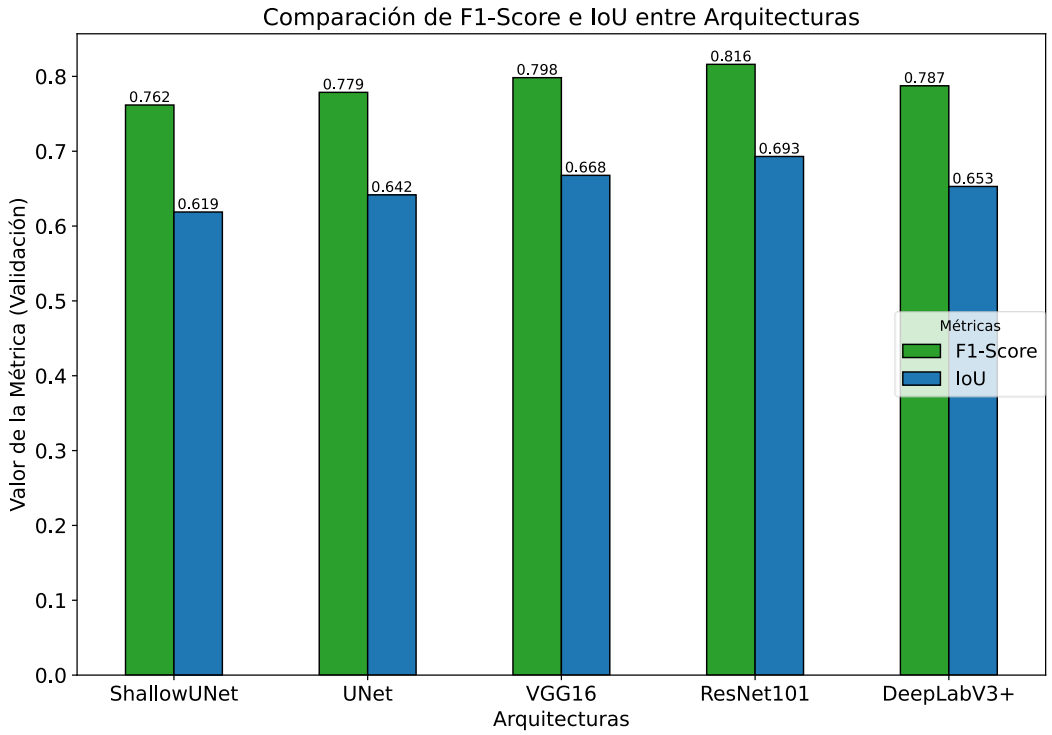


Figura 33: Gráfica de los valores en las métricas F1-Score e IoU de cada arquitectura.

La Figura 34 presenta la comparación de la pérdida en el conjunto de datos de validación entre las arquitecturas evaluadas. ResNet101 U-Net mostró la menor pérdida con un valor

de 0.0234, seguido de cerca por DeepLabV3+ que tuvo un valor de 0.0239 y VGG16 U-Net con un valor de 0.0243. Por su parte, U-Net y Shallow U-Net registraron pérdidas mayores de 0.0267 y 0.0287 respectivamente.

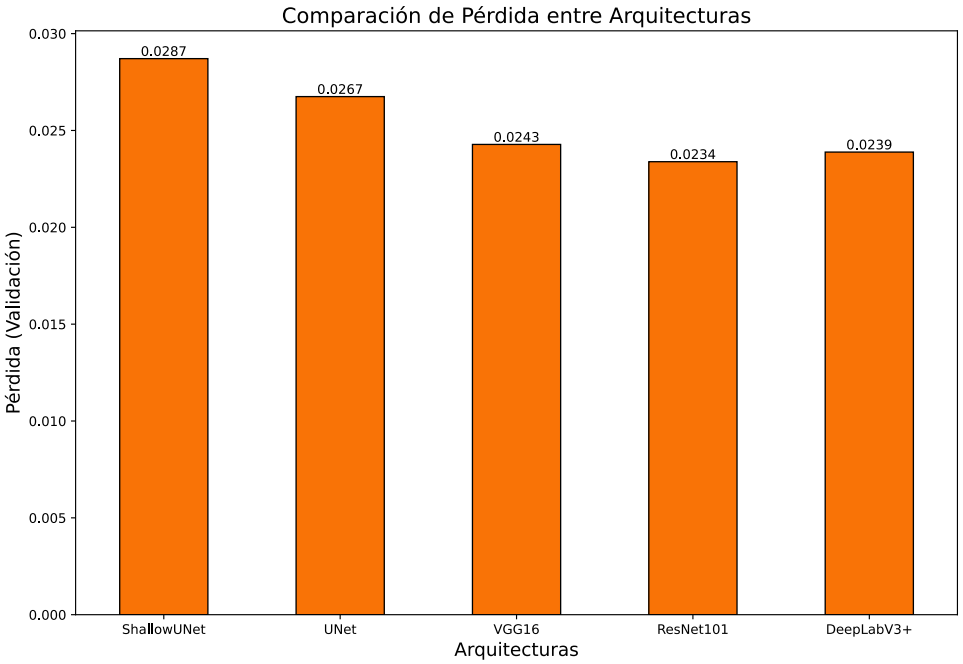


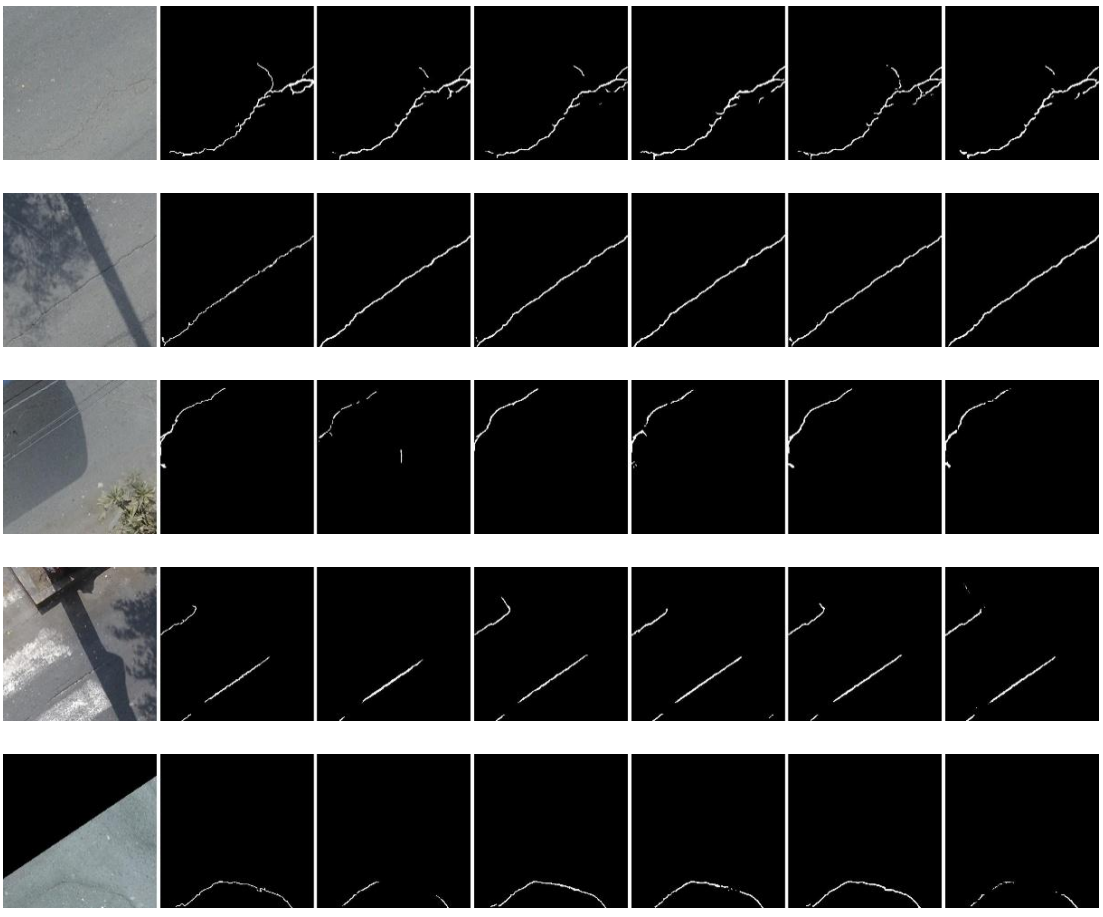
Figura 34: Gráfica de los valores de la pérdida de cada arquitectura.

Un punto interesante es que una menor pérdida no necesariamente se traduce en un mayor F1-Score o IoU, como se observa entre VGG16 U-Net y DeepLabV3+. Esto se debe a que la pérdida en validación, basada en la entropía cruzada binaria, mide la precisión de las probabilidades predichas para cada píxel en relación con las etiquetas reales, penalizando incluso pequeñas desviaciones en las probabilidades. En cambio, métricas como F1-Score e IoU se centran en la clasificación binaria de píxeles, siendo menos sensibles a pequeñas diferencias.

4.5.2.3 Comparativa de desempeño basado en la inferencia

A continuación, se presentan las inferencias realizadas sobre imágenes de prueba, seleccionando ejemplos con distintos niveles de ramificación. Aunque no se realizó una distinción específica entre los tipos de grietas durante el proceso, se seleccionaron ejemplos que representan diferentes niveles de ramificación para permitir un análisis más completo. En este análisis se consideraron tres categorías principales: grietas lineales,

caracterizadas por tener pocas o ninguna ramificación; grietas ramificadas, que presentan ramificaciones; y grietas enmalladas, cuyas ramificaciones forman patrones de polígonos cerrados. En las imágenes, la columna (a) muestra las fotografías originales, la columna (b) las máscaras de segmentación manual, los resultados de las inferencias se muestran a partir de la columna c, correspondiente a las máscaras binarias generadas por Shallow U-Net; en la columna d, las generadas por U-Net; en la columna e, las producidas por VGG16 U-Net; en la columna f, las generadas por ResNet101 U-Net; y en la columna g, las máscaras obtenidas con DeepLabv3+.



(a) (b) (c) (d) (e) (f) (g)

Figura 35: Inferencias pertenecientes a imágenes con grietas tipo lineal.

En la Figura 35 se presentan inferencias de imágenes con grietas del tipo líneal, en donde Shallow U-Net muestra limitaciones en la captura de detalles finos, omitiendo partes delgadas y mostrando un trazo más discontinuo. U-Net mejora la continuidad de las grietas, pero aún presenta algunas omisiones. VGG16 U-Net y ResNet101 U-Net destacan por su

precisión en la captura de grietas delgadas y borden bien definidos, siendo ResNet101 U-Net superior debido a la menor cantidad de omisiones. DeepLabv3+ resalta en bordes complejos, aunque puede introducir segmentaciones no deseadas en zonas de textura variada. En general, ResNet101 U-Net y DeepLabv3+ ofrecen mejores resultados para grietas lineales mientras que U-Net es una alternativa con resultados aceptables.

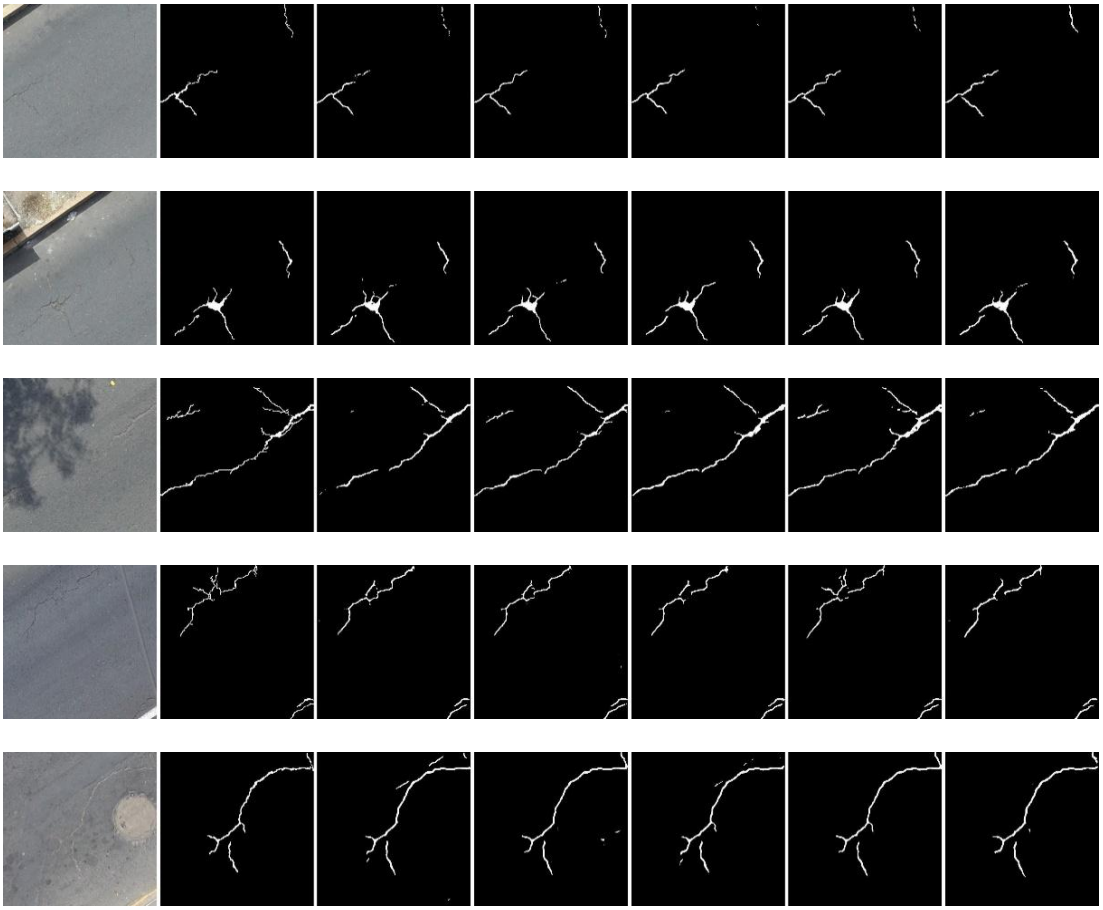
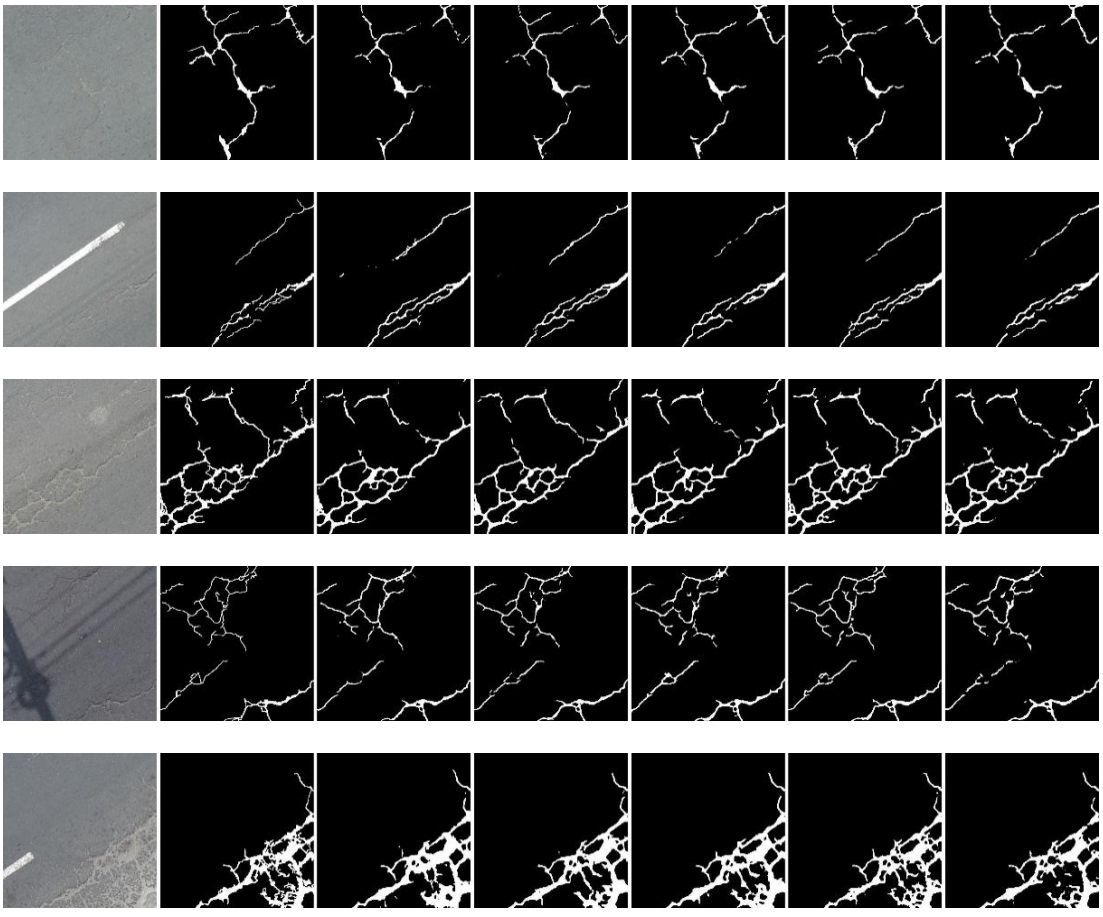


Figura 36: Inferencias pertenecientes a imágenes con grietas tipo ramificada.

En la Figura 36 se presentan inferencias de imágenes con grietas ramificadas, donde Shallow U-Net presenta dificultades para capturar la complejidad de los nodos, omitiendo conexiones y generando trazos discontinuos. U-Net mejora la representación de las ramas principales, pero presenta interrupciones en las conexiones secundarias y menores detalles en los nodos. VGG16 U-Net presenta una mejora significativa en la continuidad y

la captura en los nodos y en las ramificaciones secundarias, sin embargo, pierde precisión en las ramificaciones con grietas más finas. ResNet101 U-Net destaca en la segmentación de ramificaciones complejas y pequeñas, mostrando continuidad en los nodos, con menor pérdida de detalles. DeepLabv3+ también representa bien las ramificaciones, especialmente en las más finas y los bordes complejos, aunque puede generar segmentaciones no deseadas en zonas con texturas similares. En general, ResNet101 U-Net y DeepLabv3+ son las arquitecturas más adecuadas para este tipo de grietas, siendo superior en la segmentación de ramificaciones finas y complejas con mínima pérdida de detalles.



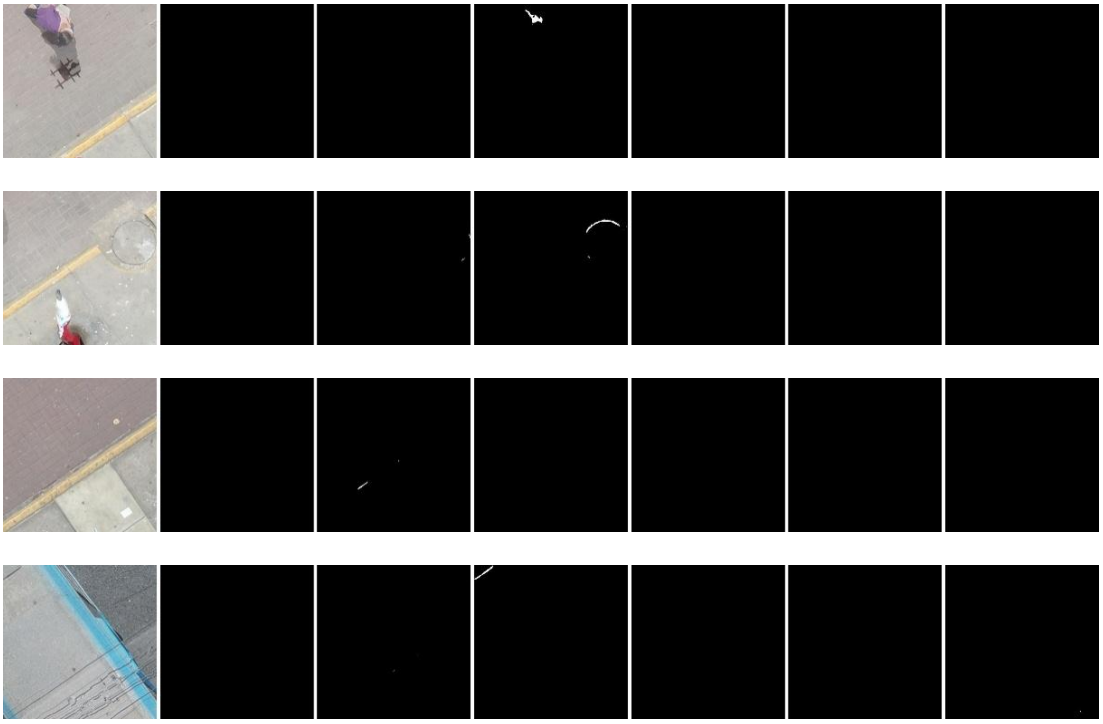
(a) (b) (c) (d) (e) (f) (g)

Figura 37: Inferencias pertenecientes imágenes con grietas tipo malla.

En la Figura 37 se presenta inferencias de imágenes con grietas enmalladas, donde Shallow U-Net presenta limitaciones al no capturar la complejidad de las intersecciones y conexiones múltiples, resultando en trazos incompletos y una pérdida de detalles en zonas

más densas. U-Net, aunque identifica mejor las conexiones principales, aún presenta discontinuidades en áreas más densas, con una representación parcial de las estructuras cerradas. VGG16 U-Net captura mejor las conexiones internas de las mallas y representa estructuras cerradas con mayor continuidad, aunque simplifica detalles en zonas más complejas. ResNet101 U-Net logra una segmentación de detalles finos y estructuras densas, logrando una representación casi completa de las mallas, con un nivel mínimo de segmentaciones no deseadas. DeepLabv3+ también logra una adecuada segmentación en áreas densas y enmalladas, destacando en la definición de bordes y detalles pequeños, aunque ocasionalmente introduce segmentaciones no deseadas en áreas con texturas similares al entorno de la grieta. En grietas enmalladas ResNet101 U-Net y DeepLabv3+ son las arquitecturas más adecuadas por su precisión en la complejidad de las conexiones y su fidelidad en la continuidad de las estructuras.

En la Figura 38 se presenta inferencias de imágenes sin grietas, donde Shallow U-Net genera algunas segmentaciones no deseadas, principalmente trazos delgados y aislados, indicando confusiones con texturas o sombras leves presentes en las imágenes. U-Net reduce estas segmentaciones no deseadas, aunque persisten pequeños trazos en áreas donde hay cambios de iluminación o contraste. VGG16 U-Net, ResNet101 U-Net destacan por generar en su mayoría, máscaras completamente negras, lo que refleja su capacidad para evitar confusiones incluso ante bordes marcados, sombras o texturas. DeepLabv3+ también produce máscaras mayoritariamente limpias, aunque en algunos casos genera trazos pequeños en áreas con sombras o líneas prominentes. En general, ResNet101 U-Net y VGG16 U-Net son las más confiables para evitar confusiones, mientras que las otras arquitecturas parecen ser más sensibles a patrones de textura o contrastes abruptos, lo que podría influir en su desempeño.



(a) (b) (c) (d) (e) (f) (g)

Figura 38: Inferencias pertenecientes a imágenes sin grietas.

4.6 Selección del modelo

Para una adecuada elección del modelo, la Figura 39 presenta un resumen de la comparación entre los modelos de segmentación evaluados, considerando su desempeño (Iou) y los costos computacionales, que incluyen el tiempo de entrenamiento (eje X), el tiempo de inferencia (eje Y) y la cantidad de parámetros (representada por el tamaño de los puntos). Donde ResNet 101-U-Net destaca por alcanzar el mejor IoU (0.69), aunque esto viene acompañado de un alto costo computacional debido a su gran cantidad de parámetros y mayores tiempos de entrenamiento e inferencia. Por otro lado, U-Net y Shallow U-Net son más eficientes en términos de inferencia, siendo Shallow U-Net una buena opción si se prioriza la rapidez, aunque con una precisión moderada. DeepLabV3+, en cambio, se posiciona como un modelo equilibrado, al ofrecer un buen IoU junto con costos computacionales razonables. Finalmente, VGG16 U-Net muestra un desempeño y costos que se sitúan en un punto intermedio, aunque no sobresale frente a las demás arquitecturas en esta evaluación.

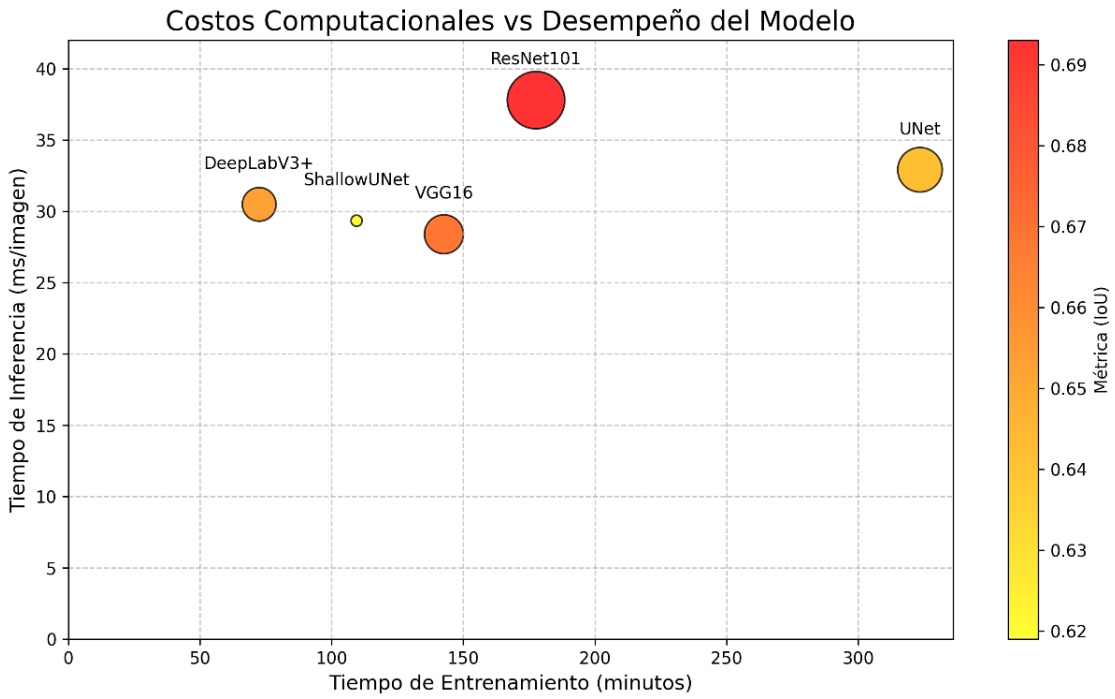


Figura 39: Gráfica resumen de costos computaciones y el desempeño del modelo.

Tras analizar los resultados obtenidos, se selecciona ResNet101 U-Net como el modelo para la segmentación de grietas debido a su destacado desempeño. Este modelo logró los valores más altos en las métricas de IoU (0.693) y F1-Score (0.816), lo que garantiza una detección de grietas precisa y confiable. Además, demostró un buen rendimiento en la identificación de diferentes tipos de grietas, incluidas las grietas lineales, ramificadas y enmalladas, así como en la distinción de imágenes sin grietas. Su capacidad para capturar detalles finos en la grieta principal, sus ramificaciones y nodos, incluso bajo cambios en la textura o el contraste de las imágenes, lo hace destacarse frente a condiciones complejas en las que otros modelos tienden a fallar o perder información. Esto asegura una evaluación integral y confiable, minimizando los errores en el análisis.

A pesar de que ResNet101 U-Net presenta un mayor tiempo de entrenamiento y de inferencia en comparación con otras arquitecturas evaluadas, estos aspectos no representan una limitación significativa en este proyecto. El tiempo de entrenamiento no influye en la decisión final, ya que es un proceso que se realiza una sola vez y no afecta el uso práctico del modelo, que ya se encuentra entrenado para su aplicación en la inferencia de ortomosaicos, lo cual se detalla en el siguiente capítulo. Por otro lado, aunque su tiempo

de inferencia es más alto, esto tampoco constituye un inconveniente, ya que el análisis no requiere resultados en tiempo real. Además, ResNet101 U-Net sigue siendo mucho más eficiente que los métodos manuales, permitiendo realizar el análisis de las vías en horas en lugar de días, lo que refuerza aún más su idoneidad para los objetivos del proyecto.

Si bien modelos como U-Net y Shallow U-Net ofrecen tiempos de inferencia más bajos, su desempeño en términos de métricas no iguala al de ResNet101 U-Net, comprometiendo la calidad de los resultados en condiciones más complejas. Por su parte, DeepLabV3+ se posiciona como una alternativa equilibrada en términos de precisión y costo computacional, pero su desempeño absoluto sigue siendo inferior al de ResNet101 U-Net.

En resumen, ResNet101 U-Net se selecciona como la mejor opción para este proyecto debido a su capacidad para proporcionar segmentaciones precisas y de alta calidad, asegurando una segmentación eficiente de las grietas en las vías. Este modelo se alinea con los objetivos del proyecto, ofreciendo una solución eficiente, rápida y más precisa que los métodos manuales, sin comprometer la calidad de los resultados.

Capítulo V: Adquisición de imágenes RPA y etiquetado manual

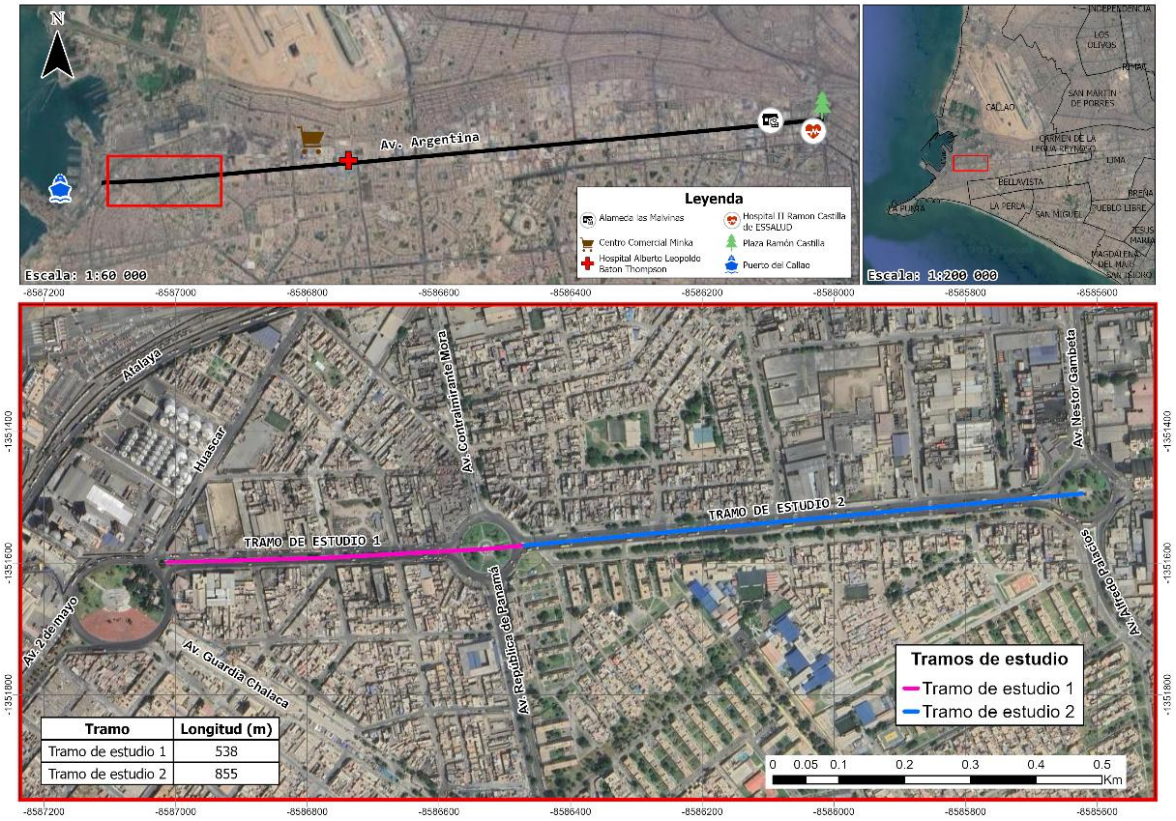
En este capítulo se aplicará el modelo seleccionado en el capítulo anterior para evaluar su efectividad en la detección de grietas en ortomosaicos generados a partir de imágenes georreferenciadas capturadas mediante dron, a través de la comparación con una segmentación manual sobre el ortomosaico en una zona representativa. Previamente, se describirá la zona donde se llevó a cabo la recolección de imágenes mediante vuelos de dron para la generación del ortomosaico, el cual servirá como base para evaluar el desempeño del modelo de segmentación, posteriormente se explicará el proceso de recolección y procesamiento de las imágenes, abarcando desde la planificación del vuelo del dron hasta la construcción de los ortomosaicos georreferenciados, asegurando que los datos sean precisos y representativos de la zona de estudio. Finalmente se segmentará de manera automatizada y manual sobre el ortomosaico para realizar una posterior comparación de esto resultados.

5.1 Tramo de estudio

Para evaluar la efectividad de modelos de *Deep learning* en la segmentación de grietas, se seleccionó la Avenida Argentina que conecta el Cercado de Lima, en la provincia de Lima, con los distritos de Carmen de la Legua y Callao, en la provincia constitucional del Callao.

Esta avenida se extiende desde el Jirón Manco Cápac, en el Callao, hasta la Avenida Alfonso Ugarte, en el Cercado de Lima, con una longitud aproximada de 10.7 kilómetros. Entre sus intersecciones más importantes se encuentran las avenidas Universitaria, Elmer Faucett, Nicolás Dueñas y Néstor Gambeta. A lo largo de su recorrido destacan puntos de interés como el Puerto del Callao, el Centro Comercial Minka, la Alameda Las Malvinas, el Hospital Alberto Leopoldo Barton Thompson, el Hospital II Ramón Castilla de ESSALUD y la Plaza Ramón Castilla.

Para evaluar la aplicación de machine learning en imágenes obtenidas mediante dron, se seleccionó un tramo de prueba de 1.4 kilómetros de esta avenida, ubicado entre las avenidas Huáscar y Néstor Gambeta, en la provincia constitucional del Callao (véase Figura 40).



Tramo	Longitud (m)
Tramo de estudio 1	538
Tramo de estudio 2	855

Figura 40: Zona de estudio. Fuente: Google Earth 2023.

5.1.1 Importancia del tramo de estudio

El tramo de la vía seleccionada desempeña un papel fundamental en la conectividad de la zona, al ser un enlace directo para diversas actividades urbanas, dado que, a una distancia aproximada de 500 metros respecto a su eje, se identificó una alta concentración de manzanas, caracterizadas principalmente por viviendas de 3 a 4 pisos. En esta misma área, se ubican también establecimientos de salud e instituciones educativas, distribuidas como se muestra en la Figura 41, lo que refleja su importancia como eje estructurador del entorno.

Aunque se trate de un tramo específico de la avenida, su función no resultó menor, ya que facilita la conexión entre numerosas viviendas, centros de atención médica e instituciones educativas ubicadas a lo largo de su recorrido. Además, su cercanía al puerto del Callao le otorga un valor estratégico ya que es uno de los puntos logísticos relevantes.

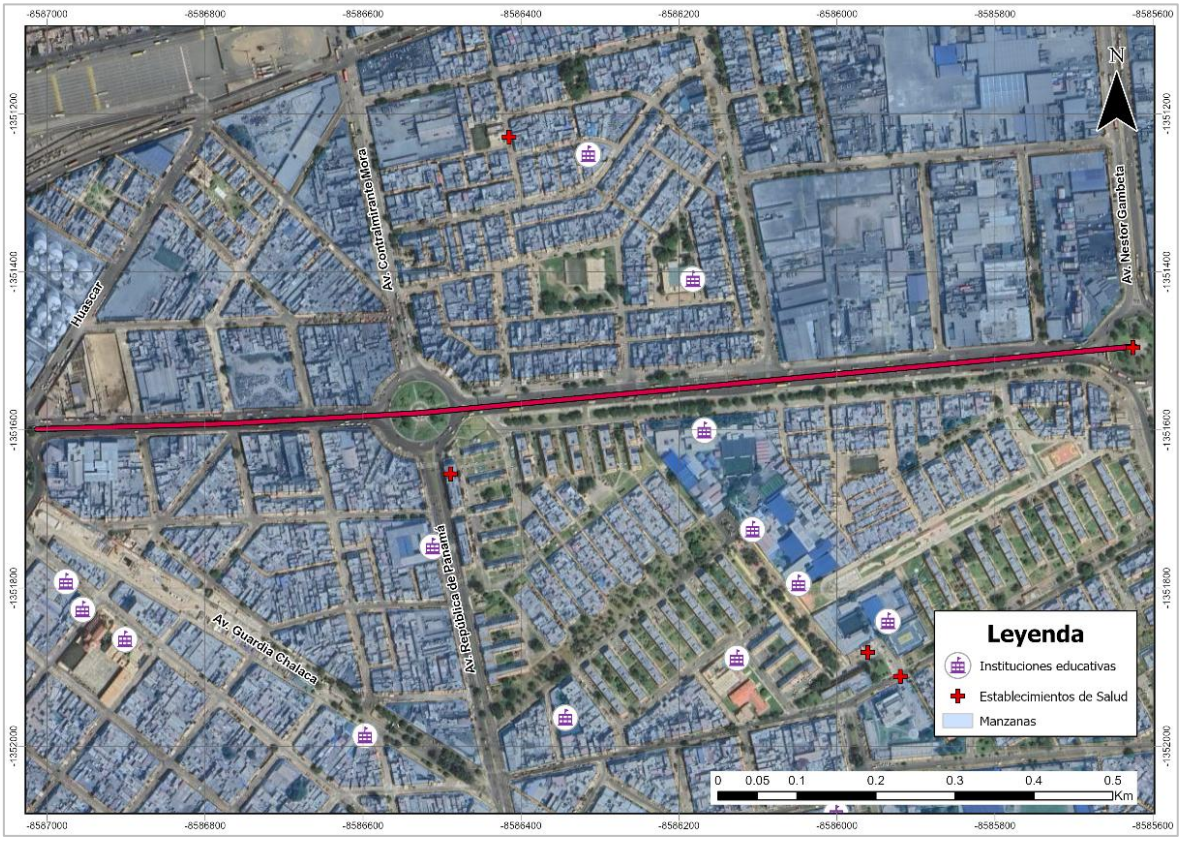


Figura 41: Distribución de viviendas, instituciones y servicios cercanos a la vía. Fuente: Google Earth – Sigrid, CENEPRED.

5.2 Recolección y procesamiento de imágenes para la generación de ortomosaicos georreferenciados

Para generar el ortomosaico de la zona de estudio, primero se recolectaron los datos en campo mediante un dron y posteriormente se procesaron en gabinete. Los detalles específicos de cada etapa, desde la captura inicial hasta la generación del ortomosaico georreferenciado, se describieron en los apartados siguientes.

5.2.1 Programación de vuelos

Previamente a la recolección de datos en campo, se llevó a cabo un reconocimiento preliminar del área utilizando Google Earth. Este análisis permitió identificar las alturas de las edificaciones y posibles obstáculos en la zona, asegurando que los vuelos se realizaran a alturas seguras y con planes de vuelo óptimos.

Los vuelos se realizaron utilizando el dron DJI Mavic 3E, cuyas características se detallaron en secciones anteriores. Dada la extensión del tramo, que abarca 1,369 m, y las limitaciones de la batería, los vuelos se dividieron en dos secciones. Además, para evaluar la influencia de la altura en la precisión de la segmentación automatizada de grietas, cada tramo se voló a alturas distintas: 35 metros para el primer tramo (tramo 1) y 30 metros para el segundo (tramo 2). Ambos planes de vuelo, con un ancho ajustado de 18 m sobre la línea de la vía, se programaron en aproximadamente 15 minutos.

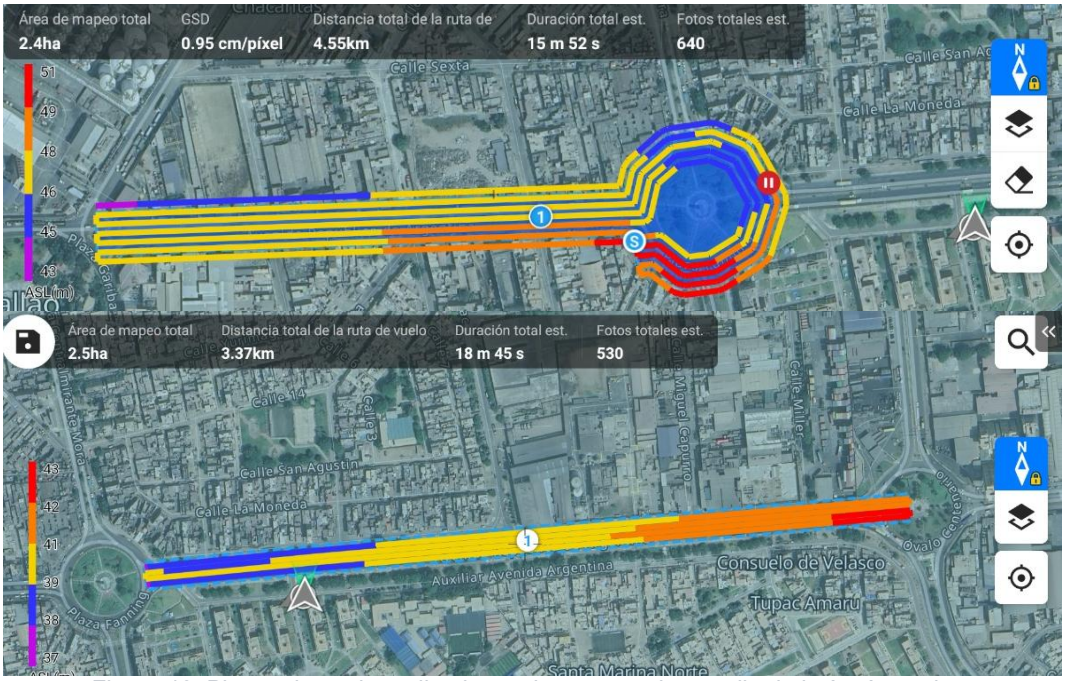


Figura 42: Planes de vuelo realizados en los tramos de estudio de la Av. Argentina.

El primer vuelo se realizó a 35 m de altura, logrando un GSD de 0.95 cm/píxel y cubriendo un área de 2.4 hectáreas, con 640 imágenes recolectadas en aproximadamente 16 minutos. El segundo se llevó a cabo a 30 m de altura, obteniendo un GSD de 0.82 cm/píxel y cubriendo un área de 5.08 hectáreas, con 530 imágenes recolectadas en

aproximadamente 19 minutos. En ambos vuelos se utilizó un solapamiento del 80% tanto longitudinal como transversal. En la Figura 42 se muestra la programación de los planes de vuelo para ambas alturas.

5.2.2 Procesamiento de datos para la generación de ortomosaicos

Las imágenes se procesaron en gabinete utilizando el software Pix4DMapper Pro, en un computador cuyas características fueron descritas en secciones anteriores. Como primer paso, se llevó a cabo un procesamiento en el que se calibraron las imágenes, obteniendo una pequeña nube de puntos, tal como se muestra en la Figura 43 y Figura 44. Esta nube de puntos permitió al software determinar las posiciones relativas de las imágenes y generar un modelo tridimensional preliminar del área, que sirvió como base para alinear las fotografías, facilitando el ajuste de parámetros geométricos y ópticos necesarios para la construcción del ortomosaico georreferenciado.

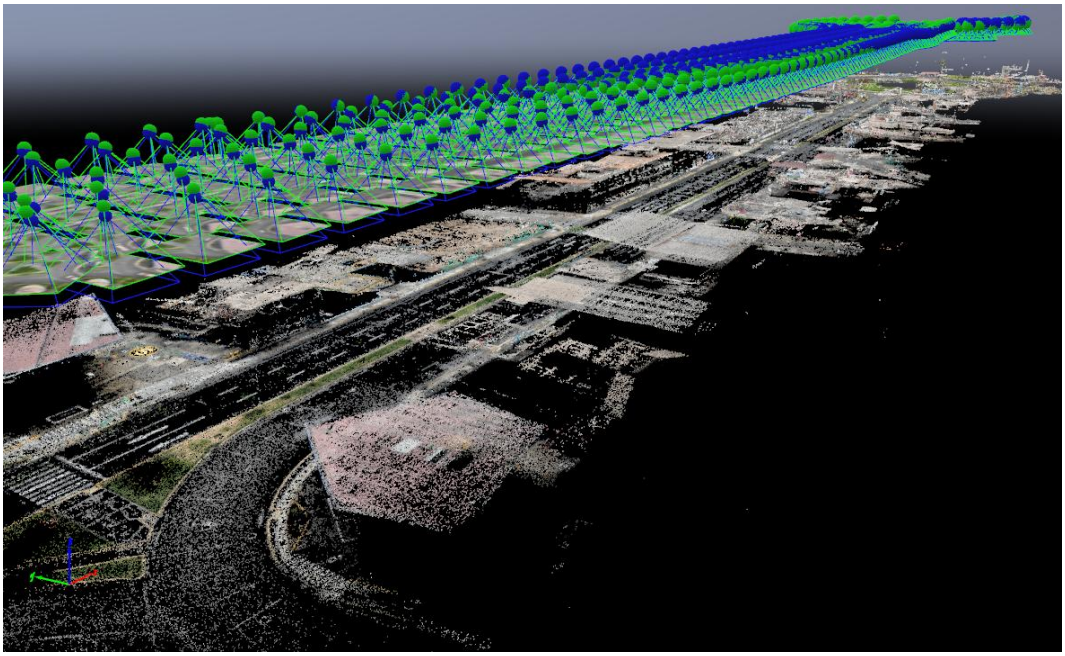


Figura 43: Nube de puntos inicial para la calibración de imágenes para el tramo 1.

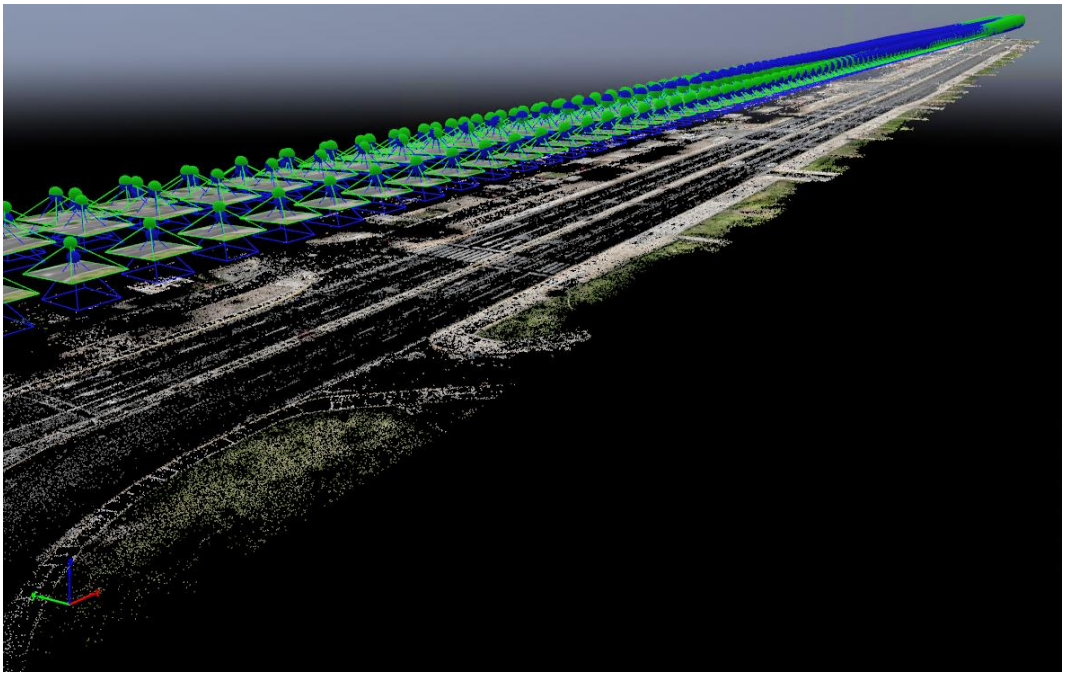


Figura 44: Nube de puntos inicial para la calibración de imágenes para el tramo 2.

A continuación, se generó una nube de puntos densa, como se muestra en la Figura 45 y Figura 46, que permitió modelar la superficie con una mayor precisión capturando características tridimensionales del terreno, proporcionando un nivel de detalle superior que será utilizado en los procesos posteriores de análisis y generación del ortomosaico.

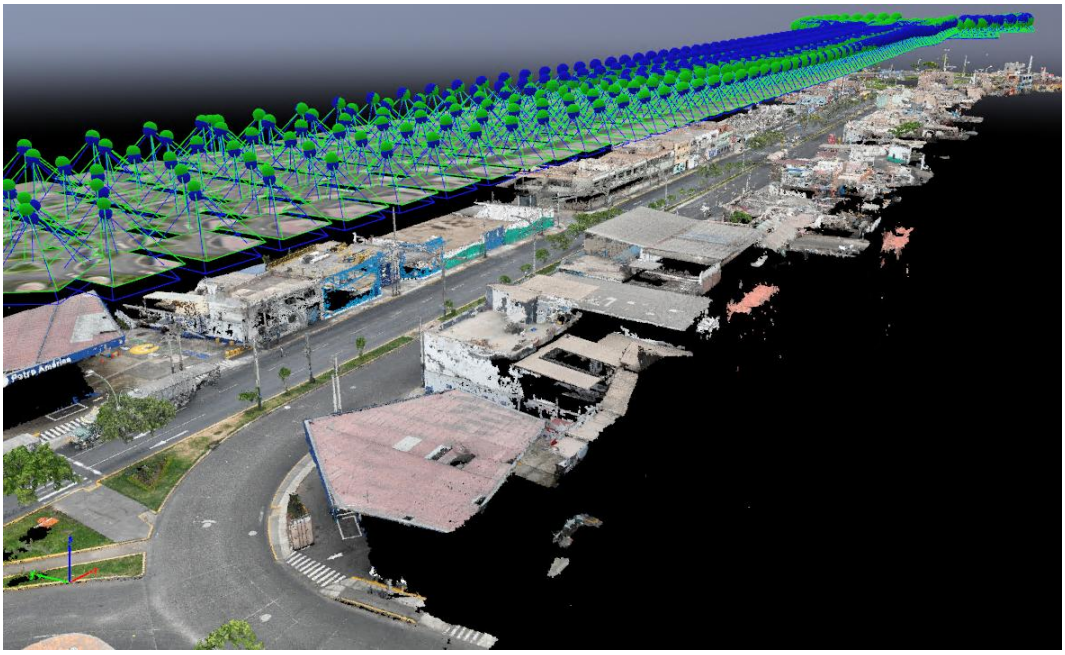


Figura 45: Nube de puntos densa para la modelación de la superficie para el tramo 1.

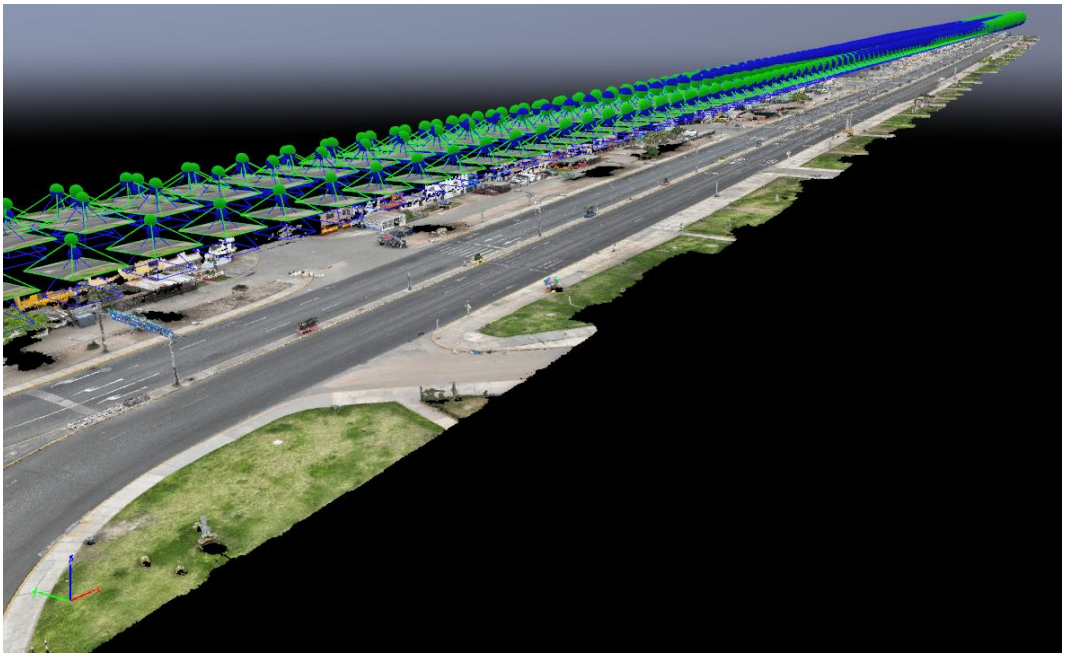


Figura 46: Nube de puntos densa para la modelación de la superficie para el tramo 2.

Finalmente, las imágenes fueron ortorrectificadas, eliminando las distorsiones causadas por la perspectiva. Este proceso dio como resultado un ortomosaico de alta resolución. Para el tramo 1 (Figura 47), con un tamaño de 65,355 × 17,379 píxeles, con un tiempo de procesamiento de 2 horas y 15 minutos. En el caso del tramo 2 (Figura 48), un tamaño de 114,185 × 17,925 píxeles, con un tiempo de procesamiento de 2 horas y 17 minutos.

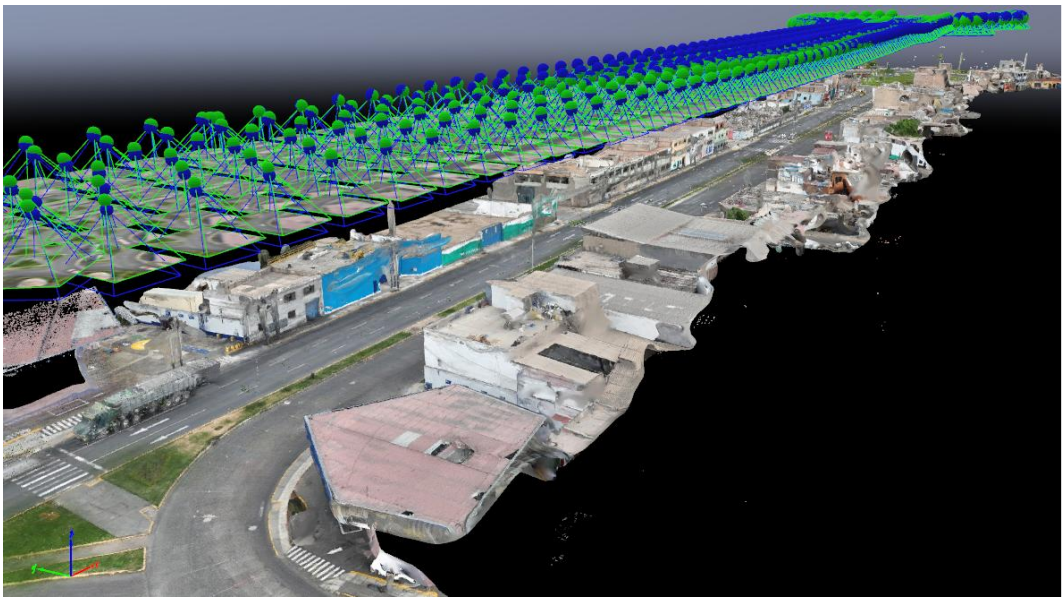


Figura 47: Triangulación de malla para la generación del modelo en 3D para el tramo 1.

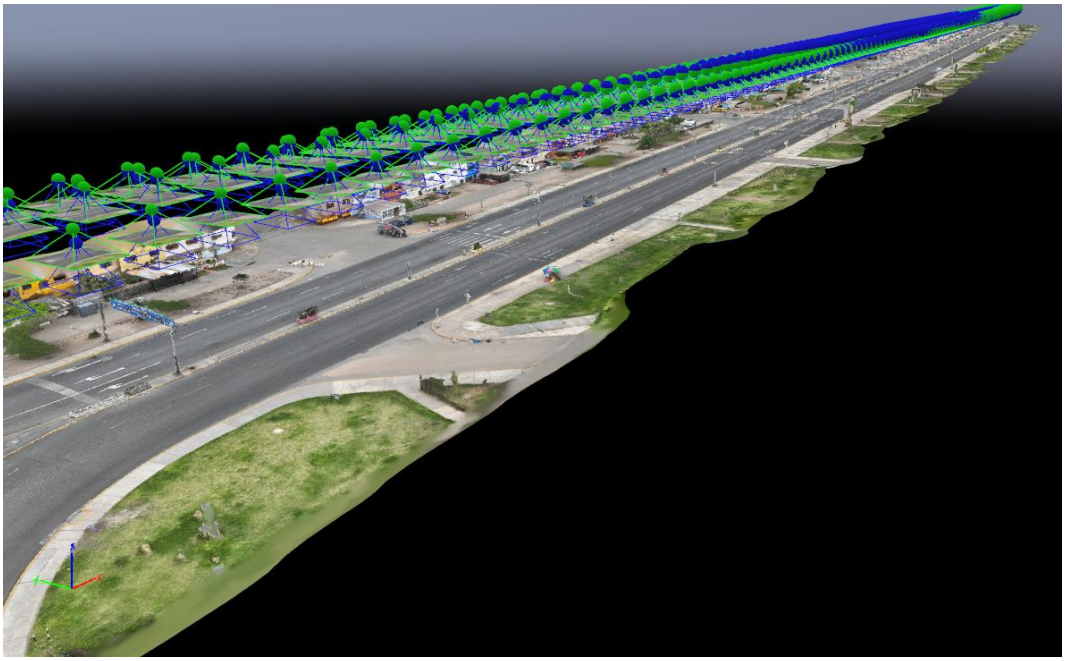


Figura 48: Triangulación de malla para la generación del modelo en 3D para el tramo 2.

Finalmente se obtuvieron los ortomosaicos para los tramos 1 y 2 como se muestra en la Figura 49.

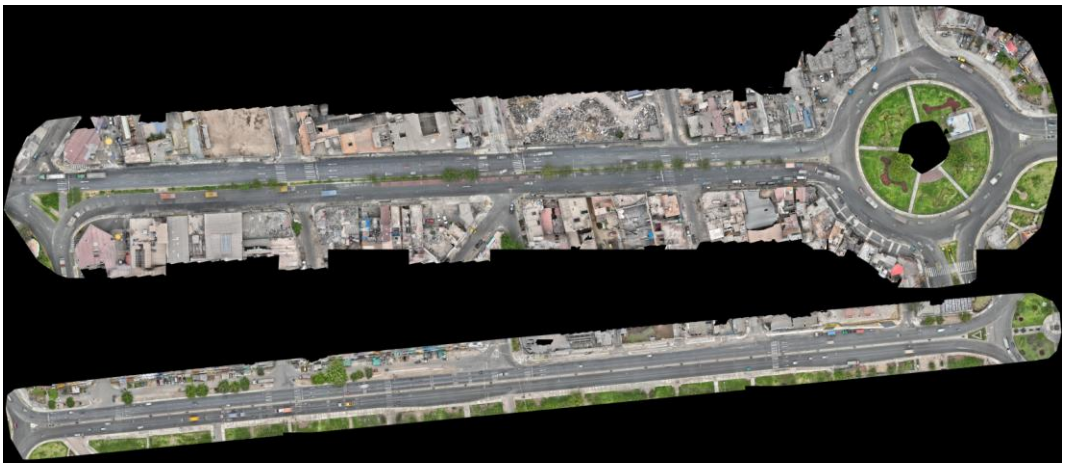


Figura 49: Ortomosaicos georreferenciado previa a la remoción de elementos en la vía para los tramos 1 y 2.

Sin embargo, se identificaron zonas donde la presencia de vehículos impedía la visibilidad de las grietas en el pavimento, como se observa en la Figura 50. Para solucionar este inconveniente, se editó la ortofoto reemplazando las áreas obstruidas, tal como se muestra en la Figura 51. En total, se modificaron 29 imágenes correspondientes al primer tramo y 35 imágenes del segundo tramo, logrando generar una ortofoto libre de obstáculos. Este proceso tomó aproximadamente 25 minutos para el tramo 1 y 30 minutos para el tramo 2, asegurando que la calidad del análisis no se viera afectada por estas interferencias.

Figura 50: Vías con vehículos que obstaculizaban la vista sobre el pavimento.

Figura 51: Vía despejada sin elementos que obstaculicen la vista sobre el pavimento.

Como resultado de todo el proceso, se generó un ortomosaico de alta resolución que representa las características de la zona de estudio, como se muestra en la Figura 52.

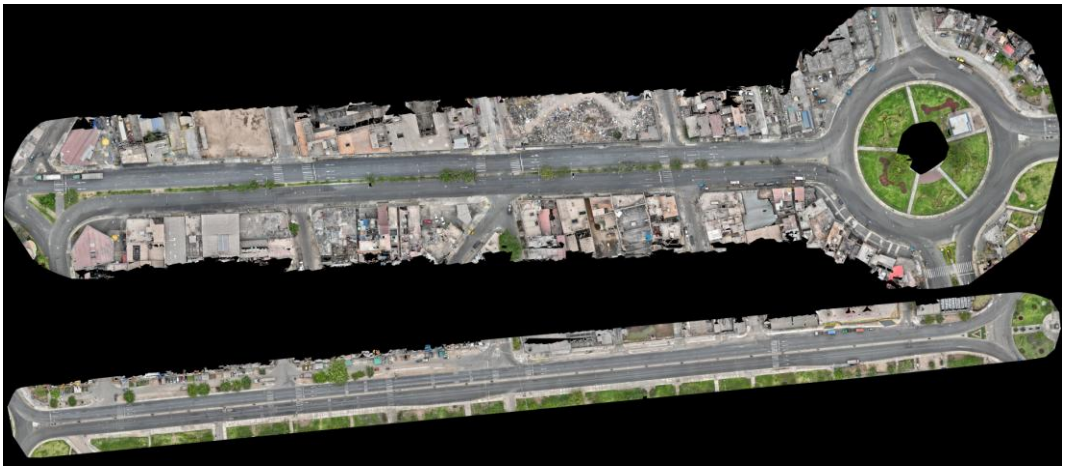


Figura 52: Ortomosaicos georreferenciado después de la remoción de elementos obstaculizadores en la vía para el tramo 1 y 2.

Este producto será empleado para las evaluaciones de segmentación de grietas, destacando la importancia de un procesamiento riguroso y detallado en la generación de datos georreferenciados de alta calidad.

A continuación, se presentan en la Tabla 8 el resumen de los tiempos en minutos para la generación de los ortomosaicos en cada uno de los tramos, desde la planificación del vuelo hasta la obtención de estos, en donde se muestra que a pesar de la diferencia entre altura de vuelos, el tiempo empleado para la generación de estos ortomosaicos son cercanos, con un diferencia de 10 minutos.

Tabla 8: Tabla resumen de los tiempos para la generación de los ortomosaicos

Datos	Tramo 1	Tramo 2
Plan de vuelo (min)	15	15
Recolección de datos (min)	16	19
Procesamiento de datos (min)	135	137
Remoción de obstáculos (min)	25	30
Tiempo total (min)	**191**	**201**

5.3 Detección de grietas en imágenes mediante inspección visual

Para la segmentación manual de grietas, se seleccionó un sector representativo cada ortomosaico, que fueron seleccionados por incluir una variedad de patrones de grietas, garantizando así un análisis comparativo más completo.

La segmentación se realizó utilizando ArcGIS Pro, empleando las herramientas de digitalización vectorial. Se trazaron manualmente las grietas visibles en el sector seleccionado, configurando un nivel de zoom óptimo que permitiera observar los detalles de las grietas, y garantizar la continuidad de las líneas.

Una vez completada, los datos vectoriales obtenidos se almacenaron en un formato geoespacial compatible, preservando su georreferenciación y facilitando su posterior comparación con las segmentaciones generadas automáticamente.

Durante la segmentación manual de los ortomosaicos, a pesar de la experiencia adquirida en la creación de la base de datos de entrenamiento, se presentaron dificultades para diferenciar con precisión las grietas del pavimento desgastado, ya que muchas de ellas tenían una apariencia similar, principalmente las grietas de menor tamaño. No obstante, este proceso resultó más sencillo en comparación con la segmentación por píxel, dado que solo fue necesario trazar líneas sobre las grietas en lugar de delimitar cada área afectada. Sin embargo, la tarea siguió siendo extenuante, ya que la delimitación solo de un sector de 76 metros de longitud en el tramo 1 y de 70 metros en el tramo 2 requirió aproximadamente 300 minutos (5 horas) en cada caso.

Los resultados de esta segmentación se pueden observar en la las Figura 56 y Figura 57 para los sectores correspondientes al tramo 1 y 2.

Capítulo VI: Comparación de la identificación de fallas detectadas con Machine Learning e inspección visual

En esta sección se describen las segmentaciones realizadas para identificar grietas en los ortomosaicos georreferenciados, utilizando la arquitectura ResNet101 U-Net, seleccionada previamente por su desempeño en la segmentación de grietas.

Además, en este capítulo se realiza una comparación entre los resultados de la segmentación manual y la obtenida mediante una inspección visual en una zona estratégicamente seleccionada de los tramos 1 y 2.

6.1 Aplicación de la arquitectura para la segmentación de grietas

El proceso de segmentación de grietas en ortomosaicos mediante *Deep Learning* se llevó a cabo en varias etapas, siguiendo una secuencia estructurada. Estas etapas, representadas en la Figura 53, abarcan desde la configuración inicial de los recursos computacionales hasta el almacenamiento de las máscaras georreferenciadas.

Figura 53: Diagrama del procedimiento estándar para la segmentación de grietas en el ortomosaico.

6.1.1 Configuración del entorno

Se llevó a cabo la configuración del entorno de desarrollo, asegurando la correcta inicialización de las librerías y dependencias necesarias. Para ello, se importaron las librerías fundamentales y otras herramientas utilizadas para la manipulación y análisis de imágenes.

Adicionalmente, se configuró el marco de trabajo de TensorFlow, garantizando su compatibilidad con las versiones de las librerías empleadas y optimizando el uso de la memoria de la GPU para mejorar la eficiencia del procesamiento.

6.1.2 Carga de datos

Para llevar a cabo la segmentación de grietas, se cargó el modelo de segmentación basado en la arquitectura ResNet101 U-Net, utilizando los pesos previamente entrenados. Durante este proceso, se verificó que el modelo estuviera correctamente configurado y listo para su uso, comprobando la compatibilidad con el entorno de ejecución.

Asimismo, se cargó el ortomosaico, almacenado en formato TIFF, para su posterior procesamiento.

6.1.3 Procesamiento de ortomosaico

Se extrajeron las dimensiones del ortomosaico, incluyendo su alto, ancho y resolución espacial, con el fin de definir la cantidad de cuadros necesarios para la segmentación. Dado que el ortomosaico se dividió en cuadros de 448 × 448 píxeles, se aplicó un solapamiento del 50% entre cuadros consecutivos, estableciendo un desplazamiento de 224 píxeles. Esta estrategia permitió una cobertura uniforme y minimizó inconsistencias en los bordes de los segmentos analizados. Ya que, si bien la inferencia puede realizarse una sola vez por recuadro, los resultados del modelo pueden variar en función del contexto circundante, pues este no procesa cada píxel de manera aislada, sino que considera la información de su entorno. Por ejemplo, un píxel ubicado en el centro de un recuadro cuenta con un contexto más amplio y definido en comparación con los píxeles cercanos a los bordes, donde la información es más limitada y puede generar predicciones menos precisas.

Además, se calcularon las posiciones de inicio y fin de cada cuadro y se eliminaron aquellos completamente negros para evitar procesamientos innecesarios.

Cada cuadro extraído fue sometido a un proceso de preprocesamiento para adecuarlo al formato requerido por el modelo. Esto incluyó la normalización de los valores de los píxeles según el esquema de ResNet101 U-Net, asegurando que las imágenes estuvieran correctamente ajustadas para su procesamiento.

Una vez preprocesados, los cuadros fueron ingresados en el modelo para la inferencia, generando una máscara de probabilidad en la que cada píxel representaba la confianza de que perteneciera a una grieta. Para el tramo 1, se realizaron inferencias en 15,037 imágenes, mientras que para el tramo 2 se procesaron 18,827 imágenes.

6.1.4 Combinación de máscaras

Una vez generadas las máscaras para cada cuadro del ortomosaico, estas fueron integradas en un lienzo global con dimensiones equivalentes al tamaño completo del ortomosaico. Dado que los cuadros extraídos presentaban un solapamiento del 50%, fue necesario gestionar la combinación de las máscaras para evitar la sobreposición excesiva de valores en las áreas de intersección, como se muestra en la Figura 54 donde la imagen de mayor tamaño representa una porción del ortomosaico, mientras que las imágenes a la derecha corresponden a los distintos recuadros que contienen la porción marcada como "5" que vendría a ser el recuadro solapado.

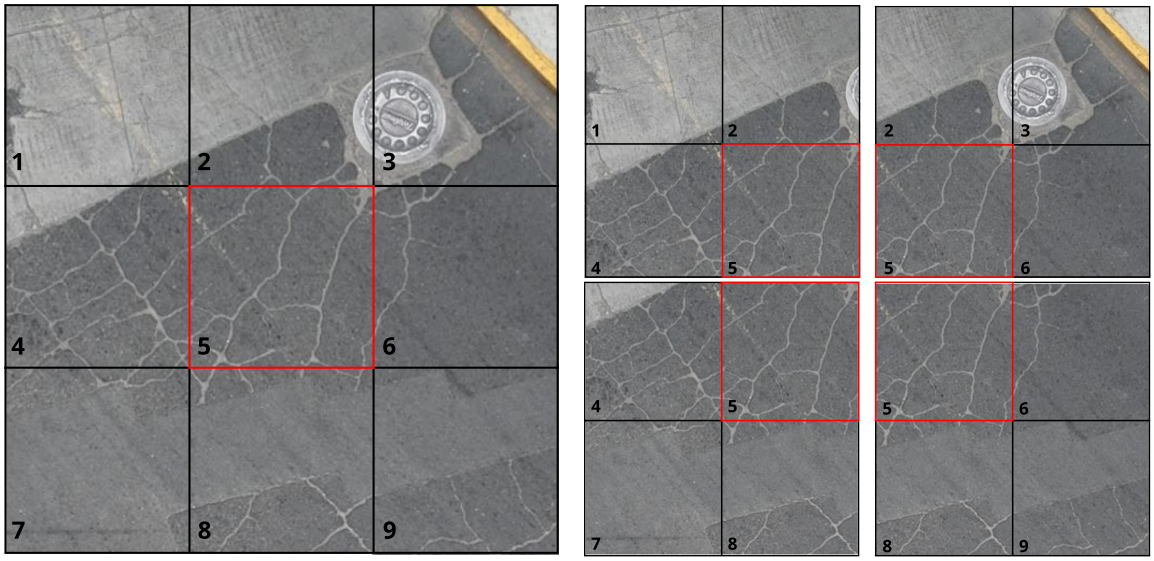


Figura 54: Recuadros de 448x448 con un recuadro de solapamiento en común de 224x224

Este solapamiento permitió que cada píxel fuera evaluado en múltiples posiciones dentro de distintos recuadros, lo que resulta en variaciones en la predicción para cada uno, como se observa en la Figura 55 donde los círculos en color blanco señalan las principales diferencias entre las inferencias que se realizaron para cada recuadro.

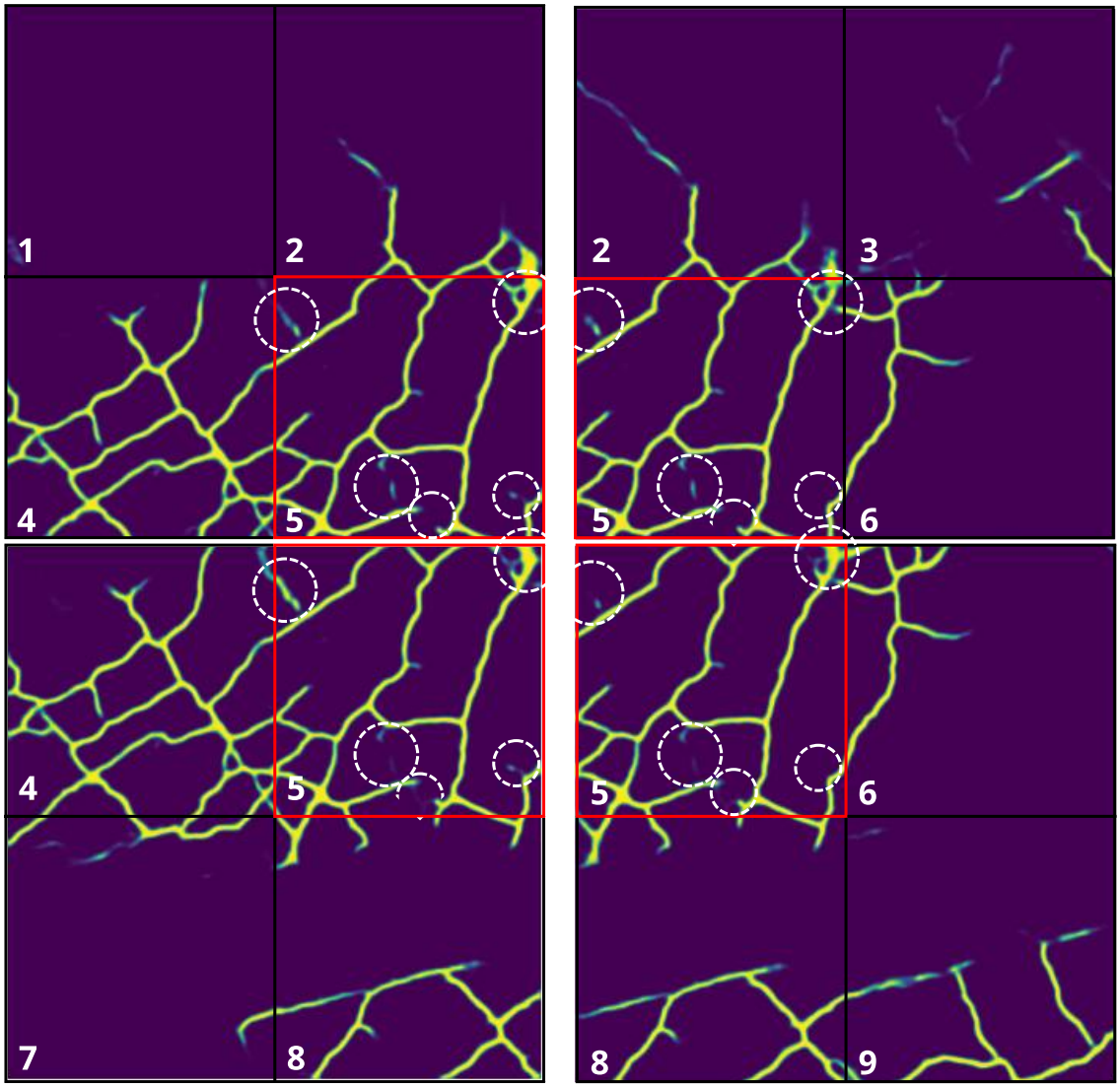


Figura 55: Máscaras de 448x448 con un recuadro de solapamiento en común de 224x224.

Para la combinación de estas máscaras se implementó un lienzo de ponderación que registró el número de veces que cada píxel fue analizado durante el proceso de inferencia. A medida que se sumaban las contribuciones de cada cuadro a la máscara global, se promediaron los valores en las áreas de solapamiento, permitiendo normalizar la superposición y distribuir equitativamente las predicciones de la segmentación en las zonas solapadas.

Al combinar diferentes estimaciones en la etapa de integración, se logró un refinamiento en la segmentación, minimizando inconsistencias en las transiciones entre cuadros y reduciendo errores en los bordes.

6.1.5 Guardado de la máscara resultante

La máscara global fue binarizada utilizando el método de Otsu, fijando un umbral mínimo de 0.01 tal y como se explicó en capítulos anteriores cuando se realizó el postprocesamiento de inferencias. Finalmente, la máscara resultante se georreferenció utilizando el archivo TIFF base, asegurando que mantuviera su alineación espacial con el ortomosaico original. Posteriormente, el archivo fue almacenado, presentando tiempos de inferencia en el ortomosaico del tramo 1 de 41 minutos y de 56 minutos para el tramo 2.

En el anexo 1, se presentan las inferencias generadas a lo largo de todo el ortomosaicos para el tramo 1 y 2. Para una mejor representación visual, se ah superpuesto la máscara georreferenciada sobre el ortomosaico correspondiente, y los resultados se muestran por sectores, lo que permite una apreciación más detallada de las características detectadas.

6.2 Comparación de resultados

Para una mejor visualización de los resultados se tomó una porción representativa de los ortomosaicos, permitiendo una comparación con la segmentación manual como muestran las Figura 56 y Figura 57 para los sectores correspondientes al tramo 1 y 2. Cada imagen incluye 3 figuras: la primera muestra el sector del ortomosaico a analizar, la segunda presenta dicho sector con la segmentación manual realizada en este capítulo, y la tercera muestra la segmentación automatizada obtenida mediante el modelo desarrollado.

Figura 56: Sector representativo del tramo 1 con segmentación manual y automatizada.

Figura 57: Sector del tramo 2 con segmentación manual y automatizada.

La segmentación automatizada para ambas alturas mostró una notable consistencia, con resultados muy similares en ambos ambos tramos. Sin embargo, se identificaron algunas fallas relevantes, principalmente en las grietas ubicadas dentro de las marcas peatonales donde el modelo no logró detectarlas correctamente. Asimismo, en ciertos tramos, la segmentación no se presentó de forma continua, evidenciándose pequeños espacios interrumpidos en algunas grietas, lo que puede considerarse un error por omisión. Por otro lado, se detectaron errores por exceso, aunque estos fueron mínimos, observándose principalmente en zonas de veredas con grietas como se observa en la Figura 58.

A pesar de estas limitaciones, la segmentación automatizada ofrece una ventaja sobre la segmentación manual, ya que proporciona una clasificación a nivel de píxel, permitiendo visualizar el ancho de las grietas. En contraste, la segmentación manual solo indica la ubicación de la grieta sin aportar información adicional sobre sus características, lo que reduce la precisión y el detalle del análisis.

Figura 58: Imagen con errores por exceso en la segmentación automatizada.

El tiempo total de la inferencia a lo largo del todo el tramo 1 fue de 41 minutos para el tramo 2 fue de 56 minutos, considerando que estos cubrían una longitud de vía de aproximadamente 550 m y 800 m, respectivamente.

Tabla 9: Comparación de segmentación manual y automatizada para cada tramo.

Tramo	Segmentación	Longitud (m)	Tiempo (minutos)
Tramo 1	Manual	76	300
	Automatizada	550	41
Tramo 2	Manual	70	300
	Automatizada	800	56

En la Tabla 9 se presenta un resumen comparativo de los tiempos entre la segmentación manual y la automatizada. Mientras que la segmentación manual tiene una velocidad aproximada de 0.24 m/min, la segmentación automatizada alcanza los 14 m/min, lo que hace que el uso de la segmentación manual resulte inviable para analizar tramos extensos. Esta diferencia resalta la ventaja de la automatización en términos de rapidez.

Conclusiones

1. Los resultados obtenidos confirman la confiabilidad de emplear modelos basados en *Machine Learning* en el procesamiento digital de imágenes capturadas por drones para la detección automatizada de fallas en el pavimento flexible. El modelo permite generar información georreferenciada en la identificación de grietas lineales, ramificadas y enmalladas, proporcionando una segmentación detallada a nivel de píxel. Asimismo, se evidenció una optimización en el tiempo de procesamiento, lo que resalta el potencial de esta tecnología como insumo inicial en inspecciones rápidas y objetivas del estado superficial del pavimento flexible. Esta información puede integrarse en diversos métodos de evaluación que consideren dichas fisuras en su catálogo de fallas, apoyando la toma de decisiones en el mantenimiento y rehabilitación de infraestructura vial.
2. La recopilación de una base de datos histórica generados a partir de 46,207 fotografías aéreas, cuyo tamaño inicial alcanzó los 30 GB y procesamiento de 1,567 imágenes de pavimento flexible, junto con la segmentación manual de fallas, permitió la construcción de una base de datos gráfica estructurada y de alta calidad. A través de técnicas de aumento de datos como rotaciones y ajustes en el brillo y contraste, se logró expandir este conjunto a 7,835, optimizando la diversidad del conjunto de entrenamiento. La distribución de los datos, con un 80% destinado al entrenamiento (6,268 imágenes) un 10% a la validación (783 imágenes) y un 10% a la prueba (783 imágenes) aseguró un entrenamiento balanceado y efectivo del modelo. Además, la adaptación de las imágenes a las condiciones del pavimento local mejoró significativamente la capacidad del modelo para identificar fallas en contextos específicos, consolidando su aplicabilidad en escenarios reales.
3. Entre las cinco arquitecturas comparadas, ResNet101 U-Net demostró el mejor rendimiento, siendo seleccionado para evaluar su efectividad en la segmentación de los tipos fallas en los ortomosaicos. La elección de este modelo se basó en los

valores de IoU y F1-score, obteniéndose valores de 0.693 y 0.816 respectivamente, que evalúan de manera precisa la calidad de la segmentación de la máscara binaria. Los tiempos de inferencia y entrenamiento no se consideraron determinantes en la evaluación, dado este trabajo de investigación no requiere segmentación en tiempo real.

4. La segmentación automatizada mediante el modelo ResNet101 U-Net demostró una alta eficiencia en la detección de fallas en el pavimento desde ortomosaicos georreferenciados, logrando resultados consistentes en comparación con la segmentación manual, reduciendo significativamente el tiempo de análisis sin comprometer la calidad general de los resultados.
5. La implementación de un traslape del 50 % durante la inferencia en ortomosaicos mejoró significativamente la continuidad de la segmentación, reduciendo errores en los bordes de los recortes y minimizando el ruido generado por predicciones erróneas.
6. El entrenamiento de los modelos de segmentación se llevó a cabo con imágenes que presentaban un GSD máximo de 1.41 cm/píxel. No obstante, el modelo demostró capacidad para detectar fallas en imágenes con valores de GSD distintos, lo que indica su adaptabilidad a diferentes resoluciones.
7. La implementación de modelos como ResNet101 U-Net en ortomosaicos georreferenciados permite integrar los resultados de la segmentación en plataformas de sistemas de información geográfica (GIS) facilitando el análisis geoespacial y la automatización de procesos en proyectos de ingeniería civil, que ayudaría de manera importante a los proyectos de mantenimiento de vías.

Recomendaciones

1. Se recomienda utilizar ortomosaicos con una resolución espacial menor a 2 cm/pixel para garantizar la precisión del modelo de segmentación y evitar valores de GSD altos que comprometerían la detección de fallas, ya que podría incrementar la cantidad de errores en la segmentación debido a la pérdida de detalle o la inclusión de ruido, afectando la calidad de la identificación de fallas.
2. Se recomienda ampliar la base de datos de entrenamiento incluyendo imágenes de diferentes: regiones geográficas, tipos de pavimento, condiciones ambientales, tipos de cámara y de drones. Esto permitirá mejorar la capacidad del modelo para generalizar y mantener un buen rendimiento en diversos contextos.
3. Se sugiere explorar arquitecturas de inteligencia artificial como los modelos basados en Transformers aplicados a visión por computadora o arquitecturas híbridas que combinen redes convolucionales (CNN) con redes neuronales recurrentes (RNN) con el objetivo de mejorar la detección de fallas en escenarios más complejos.
4. Se recomienda que futuras investigaciones se enfoquen en el desarrollo de modelos capaces de distinguir todos los tipos de fallas en el pavimento.
5. Se recomienda considerar el uso de herramientas especializadas tales como LabelMe o LabelImg para la segmentación de imágenes. Estas herramientas están diseñadas para la creación de anotaciones en tareas de visión por computadora, lo que facilitaría el etiquetado de datos de manera optimizada para su integración en modelos de aprendizaje automático.
6. Se recomienda la implementación de esta metodología en la gestión vial a nivel local, regional y nacional, así como en el sector privado, para optimizar la detección y los programas de mantenimiento de pavimentos. Además, podría incorporarse en las políticas de estado contribuyendo a la conservación y operatividad de la infraestructura vial.

7. Se recomienda realizar estudios complementarios sobre procesos probabilísticos estacionarios y su posible aplicación en el ajuste de pesos de redes neuronales convolucionales, con el objetivo de mejorar la estabilidad y generalización del modelo durante el entrenamiento.

Referencias bibliográficas

Bishop, C.M. (2006). *Pattern recognition and machine learning.* Springer

Carvajal, F., Agüera, F., & Martínez, P. (2021). *UAV photogrammetry and remote sensing.* MDPI

Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). *Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 801–818). https://doi.org/10.1007/978-3-030-01234-2_49

Chino, J., & Inchicsana, P. (2024). *Aplicación de deep learning para la detección de anomalías en pavimento en una zona del distrito de Socabaya, Arequipa 2024.* [Tesis de licenciatura, Universidad César Vallejo]. Repositorio institucional UCV. https://hdl.handle.net/20.500.12692/148029

Chollet, F. (2017). *Deep Learning with Python*. Manning Publications.

Committee on Injury, Violence, and Poison Prevention. (2009). Pedestrian safety. *Pediatrics, 124*(2), 802–812. https://doi.org/10.1542/peds.2009-1143

Coronado, J. (2002). *Manual centroamericano para diseño de pavimentos*. Secretaría de Integración Económica Centroamericana (SIECA). https://www.sieca.int/producto/manual-de-diseno-de-pavimentos/

Da Jiang Innovations (DJI). (2024). *Mavic 2 Specs*. https://www.dji.com/global/mavic-2/info

Da Jiang Innovations (DJI). (2024). *Osmo Action 4.* https://www.dji.com/global/osmo-action-4

Dirección de Tecnologías de la Información y Comunicaciones. (2021). *Anuario estadístico policial*. División de Estadística de la Policía Nacional del Perú.

https://www.policia.gob.pe/estadisticopnp/documentos/anuario-2021/anuario-estadistico-policial-2021.pdf

Dirección General de Aeronáutica Civil. (2015). *Requisitos para las operaciones de sistemas de aeronaves pilotadas a distancia. Norma Técnica Complementaria NTC 001-2015*. Ministerio de Transportes y Comunicaciones. https://www.gob.pe/institucion/mtc/informes-publicaciones/321488-ntc-001-2015-requisitos-para-las-operaciones-de-sistemas-de-aeronaves-pilotadas-a-distancia

Escalante, J., Cáceres, J., & Porras, H. (2016). Ortomosaicos y modelos digitales de elevación generados a partir de imágenes tomadas con sistemas UAV. *Tecnura, 20*(50), 119–140. https://doi.org/10.14483/udistrital.jour.tecnura.2016.4.a09

Esri. (2023). *¿Qué es la fotogrametría?.* https://pro.arcgis.com/es/pro-app/3.3/help/data/imagery/introduction-to-ortho-mapping.htm

Fernández, J., & Gutiérrez, G. (2016). Aplicaciones geológicas de los drones. *Revista de la Sociedad Geológica de España, 29*(1), 89–106. https://sge.usal.es/archivos/REV/29(1)/art6_2901.pdf

Gámiz, L. (2019). *Arquitecturas de redes neuronales profundas para mejora de voz: De los perceptrones multicapa a las redes completamente convolucionales (FCNs)* [Tesis de maestría, Universidad Autónoma de Madrid]. Repositorio institucional UAM. http://hdl.handle.net/10486/679354

García, A. (2012). *Inteligencia artificial: Fundamentos, práctica y aplicaciones*. RC Libros.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.

Ghosh, S., Chaki, A., & Santosh, K. (2021). Improved U-Net architecture with VGG-16 for brain tumor segmentation. *Physical and Engineering Sciences in Medicine, 44*(3), 703–712. https://doi.org/10.1007/s13246-021-01019-w

González, R., Ucán, J., Sánchez, I., Medina, R., Árcega, F., Zetina, C., & Casares, R. (2019). Drones. Aplicaciones en ingeniería civil y geociencias. *Interciencia, 44*(6), 326–331.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Hastie T., Tibshirani R. & Friedman J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer https://doi.org/10.1007/978-0-387-84858-7

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer. https://doi.org/10.1007/978-0-387-84858-7

Ignacio, P. (2022). *Identificación automática de las grietas en pistas de asfalto utilizando procesamiento digital de imágenes* [Tesis de licenciatura, Universidad Señor de Sipán]. Repositorio institucional USS. https://hdl.handle.net/20.500.12802/10481

Instituto Geográfico Nacional. (2016). *Fotogrametría*. https://pnoa.ign.es/

Ivanova, E., & Masarova, J. (2013). Importance of road infrastructure in the economic development and competitiveness. *Economics and Management, 18*(2), 263–274. https://doi.org/10.5755/j01.em.18.2.4253

Jiménez, J. (2000). *Visión por computador*. Paraninfo.

Jiménez, S. (2020). *Fotogrametría con drones: Conceptos y análisis*. Instituto Nacional de Investigaciones Forestales, Agrícolas y Pecuarias (INIFAP).

Jing, X., Yuan, H., Wang, Y., Huang, R., Xu, Y., & Zhang, Q. (2024). Complex crack segmentation and quantitative evaluation of engineering materials based on deep learning methods. *IEEE Access, 12*, 41396–41412. 10.1109/ACCESS.2024.3379009

Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization*. arXiv. https://arxiv.org/abs/1412.6980

Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2019). Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 9404–9413). 10.1109/CVPR.2019.00963

Kondo, Y., & Ukita, N. (2021). Crack segmentation for low-resolution images using joint learning with super-resolution. In *Proceedings of the 17th International Conference on Machine Vision and Applications (MVA)*. IEEE. https://doi.org/10.23919/MVA51890.2021.9508655

Kulkarni, S., Singh, S., Balakrishnan, D., Sharma, S., Devunuri, S., & Korlapati, S. (2022). CrackSeg9k: A collection and benchmark for crack segmentation datasets and frameworks. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 179–195). Springer Nature. 10.1007/978-3-031-25082-8_12

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

Leppich, R. (2021). *Pre-training of deep transformer encoders for time series representation models* [Tesis de maestría, University of Würzburg]. 10.13140/RG.2.2.18298.82882

Liu, Z., Cao, Y., Wang, Y., & Wang, W. (2019). Computer vision-based concrete crack detection using U-Net fully convolutional networks. *Automation in Construction, 104*, 129–139. https://doi.org/10.1016/j.autcon.2019.04.005

Ma, X. (2021). Concrete crack detection algorithm based on deep residual neural networks. *Scientific Programming, 2021*, 3137083. https://doi.org/10.1155/2021/3137083

Mei, Q., & Gül, M. (2019). A cost-effective solution for road crack inspection using cameras and deep neural networks. *Construction and Building Materials, 256*, 119397. https://doi.org/10.1016/j.conbuildmat.2020.119397

Ministerio de Transportes y Comunicaciones (MTC). (2014). *Manual de carreteras: Suelos, geología, geotecnia y pavimentos*. https://portal.mtc.gob.pe/transportes/caminos/normas_carreteras/mtc%20normas/arch_pdf/man_7%20sggp-2014.pdf

Ministerio de Transportes y Comunicaciones (MTC). (2022). *Estadística - Infraestructura de transportes - Infraestructura vial*. https://www.gob.pe/institucion/mtc/informes-publicaciones/344790-estadistica-infraestructura-de-transportes-infraestructura-vial

Montejo, A. (2006). *Ingeniería de pavimentos para carreteras*. Universidad Católica de Colombia.

Morales, M., Acosta, D., & Rojo, T. (2019). Detección de deterioros en pavimentos flexibles a partir del procesamiento de imágenes y modelos de su superficie. *Revista de Arquitectura e Ingeniería, 13*(1), 1–12. https://www.redalyc.org/articulo.oa?id=193958877006

Munro, J., Johnson, S., Cole, B., & Fisher, J. (2023). Synthetic MRI generation from CT scans for stroke patients. *BioMedInformatics, 3*(3), 791–816. https://doi.org/10.3390/biomedinformatics3030050

Orellana, M. (2019). *Detección de grietas mediante deep learning basado en imágenes en concreto* [Tesis de licenciatura, Universidad de Chile]. Repositorio institucional de la Universidad de Chile https://repositorio.uchile.cl/handle/2250/174436

Panella, F., Lipani, A., & Boehm, J. (2022). Semantic segmentation of cracks: Data challenges and architecture. *Automation in Construction, 135*, 104110. https://doi.org/10.1016/j.autcon.2021.104110.

Pazmiño, J. (2019). *Sistemas INS-GNSS RTK embarcados en drones: Comparativa y análisis de precisiones* [Tesis de maestría, Universidad Politécnica de Madrid].

Repositorio institucional de la Universidad Politécnica de Madrid. https://oa.upm.es/56183/

Putra, R., & Suryanegara, M. (2021). Performance comparison of brain tumor segmentation based on U-Net and ResU-Net architectural deep convolutional neural network. In *Proceedings of the 8th International Conference on Information Technology, Computer and Electrical Engineering (ICITACEE)* (pp. 123–126). IEEE. 10.1109/ICITACEE53184.2021.9617522

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779–788). https://doi.org/10.1109/CVPR.2016.91

Román-Garay, M., Morales-Rosales, L., Rodríguez-Rangel, H., & Fernández-Gregorio, S. (2023). Detección de deterioros superficiales en pavimentos flexibles basada en segmentación semántica y redes transformer. *Research in Computing Science, 152*(6), 61–72. https://www.rcs.cic.ipn.mx/2023_152_6/Deteccion%20de%20deterioros%20superficiales%20en%20pavimentos%20flexibles%20basada%20en%20segmentacion%20semantica%20y.pdf

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. Wells, & A. Frangi (Eds.), *Medical image computing and computer-assisted intervention – MICCAI 2015* (pp. 234–241). Springer. https://doi.org/10.1007/978-3-319-24574-4_28.

Rouhiainen, L. (2018). *Inteligencia artificial: 101 cosas que debes saber hoy sobre nuestro futuro*. Editorial Planeta.

Silva, F. (2019). *Detección automática de grietas basada en imágenes de puentes de concreto a través de modelos encoder-decoder* [Tesis de licenciatura, Universidad

de Chile]. Repositorio institucional de la Universidad de Chile. https://repositorio.uchile.cl/handle/2250/174437

Silva, L., Sanchez, H., Peral, D., Mendes, A., & Villarubia, G. (2020). An architectural multi-agent system for a pavement monitoring system with pothole recognition in UAV images. *Sensors, 20*(21), 6205. https://doi.org/10.3390/s20216205

Simonyan, K., & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition*. arXiv. https://arxiv.org/abs/1409.1556

Sola, D., & Scott, A. (2022). Efficient shallow network for river ice segmentation. *Remote Sensing, 14*(10), 2378. https://doi.org/10.3390/rs14102378

Sucar, E., & Gómez, G. (2011). *Visión computacional*. Instituto Nacional de Astrofísica, Óptica y Electrónica. https://ccc.inaoep.mx/~esucar/Libros/vision-sucar-gomez.pdf

TensorFlow. (2024). *tf.keras.callbacks.EarlyStopping*. https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping

Thomson, I., & Bull, A. (2001). *La congestión del tránsito urbano: Causas y consecuencias económicas y sociales*. Comisión Económica para América Latina y el Caribe (CEPAL). https://hdl.handle.net/11362/5674

TomTom. (2021). *Traffic index*. https://www.tomtom.com/traffic-index/ranking/

Urrego, J., & Castro, A. (2018). *Evaluación de la precisión de resultados obtenidos con el procesamiento de información generada con RPAS (Remotely Piloted Aircraft System)* [Trabajo de grado, Universidad Distrital Francisco José de Caldas]. Repositorio Institucional Universidad Distrital José Caldas. https://repository.udistrital.edu.co/items/9ef4ed13-7198-4bc1-8b3b-ecc6ad37fc99

Vásquez, L. (2002). *Pavement Condition Index (PCI) para pavimentos asfálticos y de concreto en carreteras*. Universidad Nacional de Colombia. https://sjnavarro.wordpress.com/wp-content/uploads/2008/08/manual-pci1.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems, 30*, 5998–6008. https://papers.nips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Velasco, J. (2024). *Machine learning: Fundamentos, algoritmos y aplicaciones para los negocios, industrias y finanzas*. Ediciones Díaz de Santos.

Villarroel, J. (2024). *Segmentación de cultivos de uva de mesa a partir de imágenes aéreas y satelitales usando algoritmos de aprendizaje profundo* [Tesis de licenciatura, Pontificia Universidad Católica del Perú]. Repositorio de PUCP. http://hdl.handle.net/20.500.12404/29253

Wang, P., Zhao, H., Yang, Z., Jin, Q., Wu, Y., Xia, P., & Meng, L. (2023). Fast tailings pond mapping exploiting large-scene remote sensing images by coupling scene classification and semantic segmentation models. *Remote Sensing, 15*(2), 327. https://doi.org/10.3390/rs15020327

Yang, J., & Li, V. (2020). Imaging-based crack detection on concrete surfaces using You Only Look Once network. *Structural Health Monitoring, 20*(2), 484–499. https://doi.org/10.1177/1475921720938486

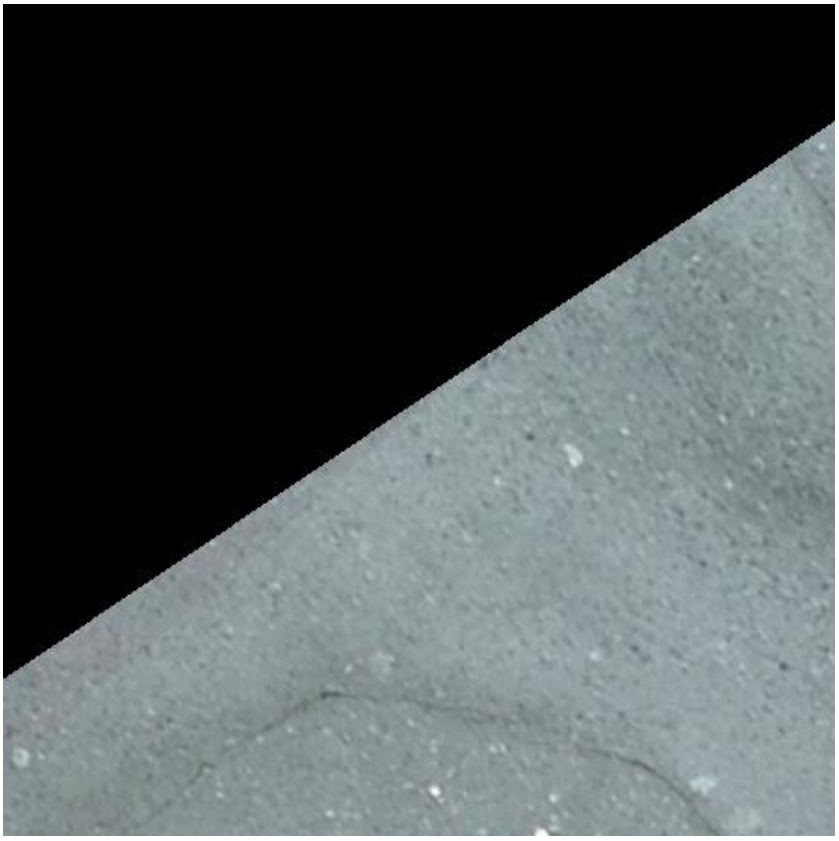
Zhang, H., Chen, N., Li, M., & Mao, S. (2024). The crack diffusion model: An innovative diffusion-based method for pavement crack detection. *Remote Sensing, 16*(6), 986. https://doi.org/10.3390/rs16060986

Zhou, Z., Siddique, M., Tajbakhsh, N., & Liang, J. (2018). UNet++: A nested U-Net architecture for medical image segmentation. In D. Shen, T. Liu, T. Peters, L. Staib, C. Essert, S. Zhou, & A. Yuille (Eds.), *Deep learning in medical image analysis and multimodal learning for clinical decision support (DLMIA/ML-CDS 2018)* (pp. 3–11). Springer. https://doi.org/10.1007/978-3-030-00889-5_1

Zúñiga, Y. (2022). *Deep learning para la detección de fallas en pavimentos de una zona del distrito de Villa María del Triunfo 2022* [Tesis de licenciatura, Universidad César Vallejo]. https://hdl.handle.net/20.500.1269

Anexos

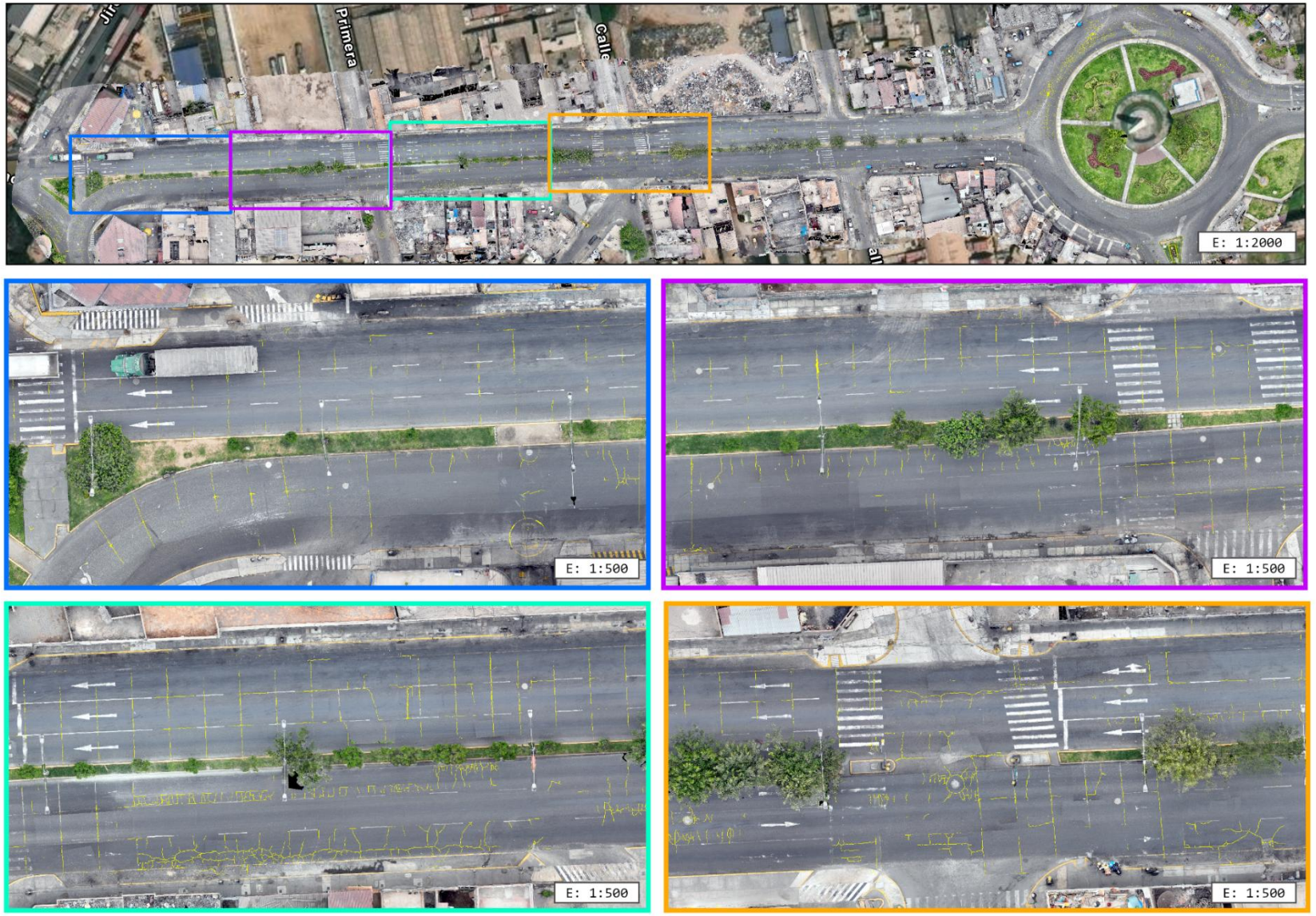
Anexo 1: Grietas Lineales



Anexo 2: Grietas Ramificadas

Anexo 3: Grietas Enmalladas

Anexo 4: Resultados de las inferencias sobre el ortomosaico



Anexo 4.1: Resultados de la segmentación automatizada sobre el ortomosaico perteneciente al tramo 1 de la Av. Argentina.

Anexo 4.2: Resultados de la segmentación automatizada sobre el ortomosaico perteneciente al tramo 1 de la Av. Argentina.

Anexo 4.3: Resultados de la segmentación automatizada sobre el ortomosaico perteneciente al tramo 1 de la Av. Argentina.

Anexo 4.4: Resultados de la segmentación automatizada sobre el ortomosaico perteneciente al tramo 2 de la Av. Argentina

Anexo 4.5: Resultados de la segmentación automatizada sobre el ortomosaico perteneciente al tramo 2 de la Av. Argentina

Anexo 4.6: Resultados de la segmentación automatizada sobre el ortomosaico perteneciente al tramo 2 de la Av. Argentina