

Desarrollo de un módulo de detección de zonas de acceso terrestre en entornos de desastres naturales mediante segmentación de imágenes aéreas.

Lara Félix, Karen Doménica

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Electrónica y Automatización

Trabajo de integración curricular, previo a la obtención del título de Ingeniera en Electrónica y Automatización

Dr. Wilbert Geovanny Aguilar Castillo

30 de julio de 2026

Certificado de análisis
Compilatio Magister+ | ESPE-ECU

MIC_Lara
ID : bf45b0e34968dcf805ced9232d25498c478df3c7

4%
Textos sospechosos

Nombre del fichero : MIC_Lara.txt
Tamaño del archivo original : 2,99 MB
Número de palabras : 19.514

Depositante : WILBERT GEOVANNY AGUILAR CASTILLO
Fecha de depósito : 30 de julio de 2026
Tipo de carga : interface
fecha de fin de análisis : 30 de julio de 2026

Resumen (sección 1/2)

Localización de los textos sospechosos en el documento :



Incluido en el porcentaje de textos sospechosos :

Similitudes <1%

Pasajes con similitudes a fuentes encontradas en diferentes colecciones.



Detección de IA 4%

Textos estilísticamente próximos a un texto generado por una IA.
Este índice es un indicador y no una prueba. Comprueba con el autor si domina los conocimientos mencionados en el documento.



No incluido en el porcentaje de textos sospechosos :

Textos entre comillas 0%

Pasajes entre comillas, a menudo indicativos de una cita.

Wilbert Geovanny Aguilar Castillo
Time Stamping Security Data

Dr. Aguilar Castillo Wilbert Geovanny

Director

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Electrónica y Automatización

Certificación

Certifico que el trabajo de integración curricular: "**Desarrollo de un módulo de detección de zonas de acceso terrestre en entornos de desastres naturales mediante segmentación de imágenes aéreas**" fue realizado por la señorita **Lara Félix, Karen Doménica,** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE; habiéndose efectuado una revisión integral de forma y de fondo conforme a los lineamientos y formatos institucionales vigentes. Además, fue revisado y analizada en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 30 de julio de 2026

Wilbert Geovanny Aguilar Castillo
Time Stamping Security Data

Dr. Aguilar Castillo Wilbert Geovanny

Director

C.C.: 0703844696

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Electrónica y Automatización

Responsabilidad de Autoría

Yo, **Lara Félix, Karen Doménica**, con cédula de ciudadanía n°1756052740, declaro que el contenido, ideas y criterios del trabajo de integración curricular**: Desarrollo de un módulo de detección de zonas de acceso terrestre en entornos de desastres naturales mediante segmentación de imágenes aéreas** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 30 de julio de 2026

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
KAREN DOMENICA LARA FELIX

Lara Félix, Karen Doménica

C.C.: 1756052740

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Electrónica y Automatización

Autorización de Publicación

Yo **Lara Félix, Karen Doménica**, con cédula de ciudadanía n°1756052740, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de integración curricular: **Desarrollo de un módulo de detección de zonas de acceso terrestre en entornos de desastres naturales mediante segmentación de imágenes aéreas** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi/nuestra responsabilidad.

Sangolquí, 30 de julio de 2026

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
KAREN DOMENICA LARA FELIX

Lara Félix, Karen Doménica

C.C.: 1756052740

Dedicatoria

A mi familia, por ser el pilar y la mayor motivación de mi vida. Gracias por impulsarme a nunca rendirme y por acompañarme en cada paso de este camino. A mi mamá, por ser mi fortaleza y por recordarme, con su incondicional "amor de amor", que siempre es posible seguir adelante. A mi papá, por ser mi guía y enseñarme con su ejemplo el valor del esfuerzo y la perseverancia. A mi hermana, por ser mi compañera de vida y estar siempre a mi lado brindándome su apoyo. A mis compañeritas de cuatro patas, Manchas Milagros y Nina, dos angelitos de luz que llenan mis días de alegría y me recuerdan el valor de los pequeños momentos.

Agradecimiento

A mi familia, por su apoyo incondicional y por brindarme siempre las oportunidades necesarias para perseguir mis sueños. Este logro también les pertenece. A la Universidad de las Fuerzas Armadas ESPE, por la formación académica y humana recibida durante estos años. Al Dr. Wilbert Aguilar por su orientación, confianza y acompañamiento durante el desarrollo de este trabajo. Finalmente, agradezco a mis amigas de la Universidad Anahí y Sara, quienes hicieron este proceso mucho más llevadero con su amistad, apoyo constante y los innumerables momentos de alegría compartidos.

Índice de contenidos

Certificación 3
Responsabilidad de Autoría 4
Dedicatoria 6
Agradecimiento 7
Resumen 14
Abstract 15
Capítulo I: Introducción 16
Antecedentes 16
Planteamiento del problema 17
Justificación 18
Alcance 19
Objetivos 21
Objetivo general 21
Objetivos específicos 21
Capítulo II: Fundamentación teórica 22
Imágenes aéreas y teledetección 22
Segmentación de imágenes 24
Redes neuronales convolucionales 27
Capas convolucionales 30
Funciones de activación 31
Batch Normalization 33
Pooling 33
Encoder y Decoder 35

Skip Connections 36
Backbones preentrenados 37
Capítulo III: Desarrollo del sistema propuesto 39
Entorno de desarrollo 39
Almacenamiento y gestión de datos: Google Drive 39
Herramienta para la edición de anotaciones: MakeSense.ai 40
Plataforma de Desarrollo: Google Colab 40
Bibliotecas 42
Base de datos 43
Construcción del conjunto de datos 44
Selección de la clase objetivo 44
Generación automática de máscaras 45
Etiquetado manual mediante MakeSense.ai 48
Generación de pseudoetiquetas 50
Dataset Final 53
Preprocesamiento de datos 53
Redimensionamiento de imágenes y Relleno (Resize with Padding) 56
Normalización 57
Binarización de máscaras 58
Generación de archivos NumPy 58
Configuración del entrenamiento 59
Preparación de los datos 59
Parámetros generales 60
Funciones de pérdida empleadas en el entrenamiento 62
Callbacks 63
Implementación de modelos 64

Implementación de U-Net....64
Implementación de LinkNet....69
Implementación de DeepLabV3+....73
Capítulo IV: Resultados y análisis....79
Métricas de evaluación del rendimiento....79
Evaluación y análisis del desempeño de los modelos....84
Evaluación cuantitativa....84
Evaluación cualitativa....88
Sistema final para la detección automática de vías de acceso terrestre....92
Discusión de resultados....94
Limitaciones del sistema....96
Capítulo V: Conclusiones y recomendaciones....97
Conclusiones....97
Recomendaciones....99
Trabajos futuros....99
Referencias:....101
Apéndices....107

Índice de tablas

Tabla 1 Bibliotecas usadas dentro de entorno de desarrollo 42

Tabla 2 Valores iniciales establecidos para los parámetros de entrenamiento de las arquitecturas 60

Tabla 3 Experimentos realizados para la optimización de U-Net. 67

Tabla 4 Configuración final del modelo U-Net 68

Tabla 5 Experimentos realizados para la optimización de LinkNet. 71

Tabla 6 Configuración final del modelo LinkNet 73

Tabla 7 Configuración final del modelo DeepLabV3+ 78

Tabla 8 Métricas de desempeño (%) 84

Tabla 9 Estadísticos de clasificación a nivel de píxel 86

Índice de figuras

Figura 1 Diagrama general del sistema propuesto para la detección de vías en imágenes aéreas 20

Figura 2 Sistema de teledetección. 23

Figura 3 Técnicas de segmentación de imágenes. 25

Figura 4 Aplicación de CNN en segmentación de imágenes. 26

Figura 5 Modelo conceptual básico de una CNN 28

Figura 6 Convolución con kernel 3x3 29

Figura 7 Stride 31

Figura 8 Gráfica de la función ReLU 32

Figura 9 Capas de Pooling 34

Figura 10 Arquitectura Encoder – Decoder. 36

Figura 11 Diagrama con skip connection 37

Figura 12 Anotaciones de clases identificadas en imágenes de conjunto de datos de FloodNet. 44

Figura 13 Generación máscara binaria a partir de JSON de dataset 46

Figura 14 Verificación generación máscara binaria a partir de JSON de dataset 47

Figura 15 Etiquetado manual en Makesense.ai 49

Figura 16 Predicción realizada de primer entrenamiento de modelo auxiliar. 51

Figura 17 Predicciones realizadas con modelo auxiliar mejorado 52

Figura 18 Flujo módulo de preprocesamiento de datos 55

Figura 19 Padding realizado a las imágenes y sus máscaras. 57

Figura 20 Arquitectura U-Net 65

Figura 21 Arquitectura LinkNet 70

Figura 22 Arquitectura DeepLabV3+.....74

Figura 23 Estructura ASPP en DeepLabV3+ 75

Figura 24 Convolución dilatada.....76

Figura 25 Matriz de confusión.....80

Figura 26 Matrices de Confusión de los Modelos de Segmentación 88

Figura 27 Predicciones de imágenes sin presencia de vías de acceso terrestre 89

Figura 28 Predicciones de imágenes con presencia de vías de acceso terrestre 90

Figura 29 Predicciones de imágenes con presencia de vías de acceso terrestre 91

Figura 30 Resultados de implementación del módulo final de detección 93

Resumen

Las inundaciones son una de las principales contingencias que comprometen la integridad de las redes viales, dificultando el acceso de los equipos de emergencia y afectando la toma de decisiones durante las operaciones de respuesta. En este contexto, el presente trabajo tuvo como objetivo desarrollar y evaluar un sistema basado en segmentación semántica para la detección automática de vías de acceso terrestre en escenarios posteriores a inundaciones mediante imágenes aéreas. Para ello, se realizó un estudio del estado del arte y se construyó un conjunto de datos binario a partir del dataset FloodNet mediante la generación y revisión de máscaras correspondientes a la clase Road Non-Flooded. Posteriormente, se aplicó un proceso de preprocesamiento que incluyó aumento de datos, redimensionamiento, normalización y partición del conjunto de datos. Sobre esta base se implementaron las arquitecturas U-Net, LinkNet y DeepLabV3+ utilizando TensorFlow/Keras, incorporando funciones de pérdida, callbacks y ajustes de hiperparámetros para optimizar el entrenamiento. El desempeño de los modelos fue evaluado mediante métricas de segmentación y un análisis cuantitativo y cualitativo de las predicciones obtenidas. Los resultados evidenciaron que DeepLabV3+ alcanzó el mejor desempeño, con un IoU del 84,35 %, un coeficiente Dice del 91,51 % y una precisión del 93,97 %, además de reducir aproximadamente un 45 % los falsos positivos respecto a U-Net y LinkNet. Se concluye que la metodología propuesta constituye una alternativa efectiva para la identificación automática de vías de acceso terrestre en escenarios post inundación, proporcionando información que puede contribuir a una planificación más eficiente de las operaciones de respuesta ante emergencias.

Palabras clave: segmentación semántica, visión por computador, inteligencia artificial, inundaciones, FloodNet.

Abstract

Floods are one of the main contingencies that compromise the integrity of road networks, hindering access for emergency teams and affecting decision-making during response operations. In this context, this work aimed to develop and evaluate a semantic segmentation-based system for the automatic detection of land access routes in post-flood scenarios using aerial imagery. To this end, a state-of-the-art review was conducted, and a binary dataset was constructed from the FloodNet dataset by generating and reviewing masks corresponding to the Road Non-Flooded class. Subsequently, a preprocessing process was applied, including data augmentation, resizing, normalization, and partitioning of the dataset. Based on this, the U-Net, LinkNet, and DeepLabV3+ architectures were implemented using TensorFlow/Keras, incorporating loss functions, callbacks, and hyperparameter adjustments to optimize training. The performance of the models was evaluated using segmentation metrics and a quantitative and qualitative analysis of the predictions obtained. The results showed that DeepLabV3+ achieved the best performance, with an IoU of 84.35%, a Dice coefficient of 91.51%, and an accuracy of 93.97%, in addition to reducing false positives by approximately 45% compared to U-Net and LinkNet. It is concluded that the proposed methodology constitutes an effective alternative for the automatic identification of land access routes in post-flood scenarios, providing information that can contribute to more efficient planning of emergency response operations.

Keywords: semantic segmentation, computer vision, artificial intelligence, flooding, FloodNet

Capítulo I: Introducción

En el presente capítulo se define el punto de partida de la investigación de manera que se delimita el problema de estudio y se presenta una propuesta de solución. Para esto se exponen los antecedentes y planteamiento del problema que se ha identificado además de la respectiva justificación que demuestra que el presente trabajo de integración curricular es viable de realizar. Finalmente, se definen el alcance y los objetivos que guiarán el desarrollo de este trabajo.

Antecedentes

Los desastres naturales representan una de las principales causas de afectación a la infraestructura vial, generando daños importantes en carreteras y puentes, de forma que llegan a dificultar o impedir el acceso oportuno de equipos de rescate a las zonas afectadas. En este sentido, las inundaciones ocasionan grandes alteraciones en cuanto a la movilidad debido a los significativos cambios en las condiciones del terreno, vías completamente saturadas de agua o en su defecto de escombros, limitando las labores de evaluación inicial de daños y respuesta oportuna tras las primeras horas posteriores al evento (Patiño Duarte et al., 2026). Por lo tanto, una identificación rápida de vías accesibles es fundamental para una eficaz coordinación de las respectivas operaciones de emergencia.

Tradicionalmente, los daños provocados por los desastres naturales han sido evaluados mediante inspecciones realizadas directamente en campo, implicando una gran inversión de tiempo, así como de recursos humanos, sin tomar en cuenta la exposición del personal a condiciones peligrosas. Sin embargo, en los últimos años se ha implementado el uso de vehículos aéreos no tripulados (UAV) para capturar rápidamente imágenes de alta resolución de las zonas que han sido afectadas por los desastres, permitiendo obtener información visual detallada del estado de la infraestructura terrestre sin requerir una inspección in situ en la zona de emergencia (Rahnemoonfar et al., 2020).

En este contexto, la visión por computador junto con la inteligencia artificial está tomando un papel importante en el análisis automatizado de imágenes aéreas. Específicamente, la segmentación semántica basada en redes neuronales convolucionales destaca por su alta capacidad para reconocer precisamente elementos presentes en una escena incluyendo masas de agua, vías, construcciones, entre otros (Fawakherji et al., 2025). Sin embargo, se evidencia en la literatura que son limitados los estudios relacionados específicamente a la detección automática de caminos viables en escenarios post inundación, lo que motiva al desarrollo del presente trabajo de integración curricular.

Planteamiento del problema

Las inundaciones son uno de los desastres naturales que generan mayor impacto en la infraestructura vial, ya que dificultan la movilidad y afectan la accesibilidad dentro de las zonas perjudicadas. Esta situación impide una atención oportuna a las personas afectadas por el desastre y retrasa la ejecución de las operaciones de respuesta ante este tipo de emergencias (Muñoz Gómez, 2025).

Ante esta situación, la evaluación de las zonas afectadas depende en su mayoría de inspecciones en campo, registros fotográficos y elaboración de mapas de riesgo tras realizar recorridos manuales con el objetivo de identificar las áreas críticas. Sin embargo, a pesar de los valiosos datos que proporcionan las actividades mencionadas, estas requieren un importante despliegue de personal y horas de trabajo, debido a la gran cantidad de información visual recopilada durante el desastre que debe ser analizada para una identificación de vías potencialmente transitables. (Sánchez Álvarez, 2024). La demora que conlleva realizar este procedimiento manual no es óptima en situaciones de emergencia en donde es imprescindible contar con una rápida planificación de las operaciones de atención y rescate.

En los últimos años, los avances de visión por computador e inteligencia artificial han fomentado el desarrollo de modelos de segmentación de imágenes basados en redes neuronales convolucionales capaces de procesar datos visuales tal como imágenes aéreas

obtenidas mediante vehículos aéreos no tripulados (UAV) para así analizarlas de manera automática, facilitando tareas como la clasificación de imágenes, detección de objetos y evaluación de daños posteriores a desastres naturales. En ese sentido, se presenta el conjunto de datos FloodNet 2021, mismo que ha impulsado estas investigaciones ya que promueve tareas de segmentación semántica de imágenes de alta resolución incorporando anotaciones de elementos como edificios, vegetación, cuerpos de agua, vehículos, carreteras inundadas o no inundadas (Rahnemoonfar et al., 2020). Sin embargo, la literatura continúa con su enfoque principal en identificación de objetos, existiendo un campo de estudio limitado en cuanto a la detección automática de caminos transitables para el auxilio en emergencias, un aspecto clave para optimizar la toma de decisiones en campo.

Debido a la problemática expuesta, existe la necesidad de desarrollar un módulo de detección de vías de acceso terrestre transitables para vehículos de respuesta ante emergencias mediante el análisis de imágenes aéreas en escenarios de inundación. Por consiguiente, el presente proyecto se centra en la identificación de caminos no inundados mediante la evaluación del desempeño de diferentes arquitecturas de segmentación semántica a fin de determinar la alternativa con mejores resultados para resolver este problema.

Justificación

Durante una inundación, conocer qué vías permanecen transitables define el éxito de la respuesta ante la emergencia. Como ya se mencionó con anterioridad, el método tradicional de inspección en terreno retrasa significativamente el despliegue de ayuda porque el acceso físico en campo puede ser limitado, peligroso y el tomar dichos datos, procesarlos y analizarlos puede tomar varias horas. Automatizar el análisis de imágenes obtenidas por drones resuelve este problema al entregar mapas de rutas accesibles de forma inmediata. Es así que, los equipos de rescate contarían con una mejor información para planificar evacuaciones en condiciones seguras y optimizando los recursos empleados para las mismas. La necesidad de

esta herramienta se evidencia en la provincia de Imbabura, donde el Servicio Integrado de Seguridad ECU 911 reportó que se tuvo cerca de diez mil emergencias relacionadas con lluvias e inundaciones en el lapso de un año (ECU 911, 2024). A pesar de que estas instituciones siguen coordinando su respuesta ante emergencias en base a la inspección visual manual, es importante destacar que incorporar el análisis automatizado que se presenta en esta investigación aceleraría el trazado de mapas en las zonas impactadas.

Frente a este escenario, en el presente trabajo se busca evaluar el uso de segmentación semántica para detectar vías de acceso terrestre automáticamente en base a la comparación del entrenamiento de tres modelos de redes neuronales convolucionales como lo son U-Net, LinkNet y DeepLabV3+, que se seleccionaron debido a su gran capacidad probada para la clasificación de imágenes pixel a pixel (Ronneberger et al., 2015). Para esto, el sistema tomará un conjunto de imágenes de alta resolución capturadas por drones. Mediante esta propuesta, se supera la limitación de los sistemas de monitoreo visual tradicionales, gracias a la integración de herramientas avanzadas de inteligencia artificial para optimizar la gestión operativa de desastres.

Alcance

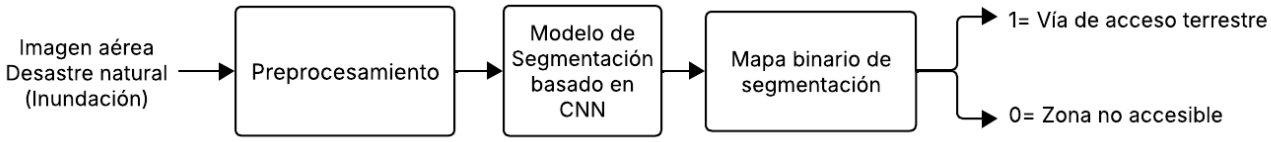
El presente trabajo de integración curricular tiene como alcance el desarrollo de un módulo de software basado en el uso redes neuronales convolucionales (denominadas CNN) para realizar la segmentación de imágenes aéreas, orientado a la detección automática de vías terrestres en escenarios de inundación. Para ello, el proyecto se limita al uso de bases de datos de acceso abierto (Open Access) que contengan imágenes previamente etiquetadas. Además, el trabajo contempla la selección, limpieza y preprocesamiento del dataset, a fin de que todas las imágenes se ajusten a los parámetros requeridos para el entrenamiento de los tres modelos de CNN.

Como se ilustra en el diagrama general de la Figura 1, se plantea un problema de segmentación estrictamente binario. El módulo generará un mapa binario de accesibilidad

terrestre, dando como resultado dos únicas posibles categorías: las vías de acceso terrestre potencialmente transitables que son asignadas a la clase 1, y las zonas no accesibles, representadas por la clase 0. No se realizará identificación o clasificación de otros elementos tales como masas de agua, vegetación o edificaciones, ya que el propósito es solo identificar vías aptas para el desplazamiento ágil y seguro de vehículos de rescate y las unidades de soporte en emergencias.

Figura 1

Diagrama general del sistema propuesto para la detección de vías en imágenes aéreas



Nota. Elaboración propia

En cuanto al desarrollo del módulo, se programarán y compararán tres arquitecturas de redes neuronales convolucionales: U-Net, LinkNet y DeepLabV3+. Se evaluará el desempeño de las mismas mediante métricas cuantitativas de clasificación binaria, dando prioridad a sensibilidad y especificidad, índice de intersección sobre la unión, precisión y matriz de confusión.

Finalmente, es importante señalar que el presente trabajo no considera la implementación del software en tiempo real ni en plataformas físicas o uso de hardware especializado por lo que todo se desarrollará en un entorno de simulación controlado. Adicional, no se realizará procesos posteriores como la planificación de rutas, navegación autónoma o toma de decisiones logísticas en zonas afectadas ya que eso corresponde a futuras líneas de investigación.

Objetivos

Ante lo ya expuesto, el desarrollo de este trabajo de integración curricular va orientado al diseño, entrenamiento y evaluación de un módulo de software que tenga la capacidad de procesar información visual aérea y de esta manera facilite la toma de decisiones logísticas por parte de los equipos de emergencia en situaciones de desastres naturales como lo son las inundaciones.

Objetivo general

Desarrollar un módulo de detección de zonas de acceso terrestre en entornos de desastres naturales mediante segmentación de imágenes aéreas.

Objetivos específicos

- Realizar el estudio del estado del arte sobre segmentación de imágenes aplicada a la detección de caminos, zonas transitables y áreas afectadas en escenarios de desastres naturales.
- Diseñar el esquema de segmentación de imágenes aéreas, de las bases de datos, para la identificación de zonas de acceso terrestre, considerando distintos tipos de terreno y condiciones de daño.
- Implementar modelos de segmentación basados en técnicas de visión por computador e inteligencia artificial para la detección automática de zonas accesibles.
- Evaluar la precisión y robustez del módulo de segmentación en escenarios controlados.

Capítulo II: Fundamentación teórica

El fundamento teórico constituye la base que orienta el desarrollo de una investigación, en el que se distingue teorías, conceptos y aportes científicos que permiten la comprensión integral del problema de estudio desde una perspectiva técnica y metodológica (Bernal, 2010). En este contexto, la revisión de literatura especializada proporciona los criterios necesarios para interpretar los principios relacionados con la adquisición y procesamiento de imágenes aéreas, además del funcionamiento de modelos de segmentación basados en redes neuronales convolucionales.

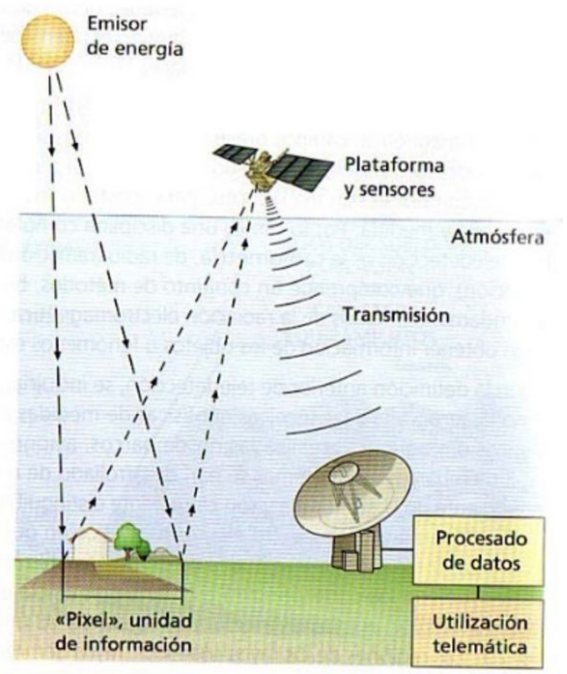
Por consiguiente, los contenidos desarrollados en el presente capítulo sustentan las decisiones adoptadas durante la investigación y significan el soporte conceptual para el desarrollo del módulo de detección de zonas de acceso terrestre en entornos de desastres naturales, específicamente en inundaciones.

Imágenes aéreas y teledetección

La teledetección es una técnica mediante la cual se obtiene información de la superficie terrestre sin establecer contacto directo con ella, más bien, a partir de la captación de la radiación electromagnética mediante sensores especializados, tal como se visualiza en la Figura 2. De esta manera la información adquirida puede emplearse para la generación de mapas temáticos y cartográficos, facilitando el análisis de las características del terreno, la cobertura del suelo y otros elementos geográficos de interés. Por lo cual, gracias a estas capacidades, la teledetección se ha consolidado como una herramienta de apoyo para el monitoreo del territorio y la obtención de información espacial en diferentes ámbitos de aplicación (Arozarena Villar, n.d.).

Figura 2

Sistema de teledetección.



Nota. El gráfico presenta los componentes de un sistema de teledetección. Tomado de (Redondo Carpio, 2024).

De igual manera, entre las plataformas utilizadas para la adquisición de información mediante teledetección destacan los vehículos aéreos no tripulados (UAV, por sus siglas en inglés), que integran sensores capaces de capturar fotografías y vídeos con una alta resolución. A través de estas imágenes es posible la aplicación de técnicas de fotogrametría digital para generar ortomosaicos, realizar mediciones aéreas y elaborar cartografía de áreas urbanas o rurales con un alto nivel de detalle. En esta línea, los UAVs se convierten en una alternativa eficiente para obtener información actualizada de zonas de interés en tiempos reducidos (Sani et al., 2015).

En escenarios de desastre, la identificación de vías transitables se vuelve un elemento clave para facilitar las operaciones de búsqueda, rescate y distribución de ayuda humanitaria. Sin embargo, las condiciones de terreno y las restricciones de acceso pueden dificultar la

evaluación directa de las zonas afectadas, aumentando el tiempo empleado para la toma de decisiones. Bajo este contexto, los UAV permiten obtener información aérea de manera rápida y segura, reduciendo los tiempos de inspección y proporcionando imágenes que simplifiquen la identificación de rutas de acceso para los equipos de respuesta (Díaz Olariaga, 2024).

Segmentación de imágenes

Dentro del campo del procesamiento digital de señales y visión por computador, la manera de interpretar la información visual varía de acuerdo a la aplicación que se requiera. En este sentido, los tres principales trabajos realizados son la clasificación de imágenes, detección de objetos y segmentación. La clasificación determina la presencia de un objeto en una imagen general mientras que la detección busca un objeto y lo encuadra en una caja delimitadora conocido como *bounding box.* Por otro lado, la segmentación analiza las imágenes a nivel de píxeles permitiendo trazar con exactitud los límites de objetos dentro de una imagen. (García Vega, 2024)

Para el presente proyecto se trabajará específicamente en tareas de segmentación de imágenes, que tiene tres variantes principales que se podrán identificar en la Figura 3. A continuación, se explica los tipos de segmentación:

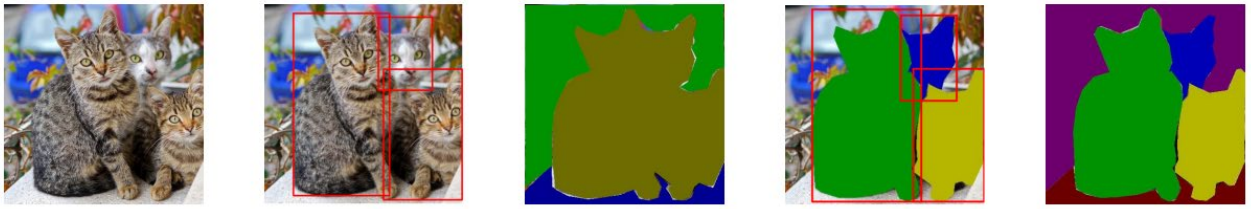
- Segmentación Semántica: Este tipo de segmentación consiste en asociar cada píxel de la imagen a una clase o categoría específica, sin distinguir entre instancias individuales de un mismo grupo (Miguélez, 2022), tal como se aprecia en la Figura 3(c), en donde se visualiza que todos los elementos correspondientes a la clase ‘gato’ se pintan de un mismo color.
- Segmentación de Instancias: Consiste en la combinación de detección de objetos y segmentación semántica, con el objetivo de encontrar y marcar cada objeto de manera independiente dentro de una imagen, así estas pertenezcan al mismo tipo (Miguélez, 2022). Esto se puede observar en la Figura 13(d), donde se aplican los *bounding boxes* específicamente a la categoría de gatos, sin

embargo, a su vez se puede visualizar que cada gato tiene colores diferentes, lo que indicaría que cada objeto tiene un valor distinto.

- Segmentación Panóptica: Esto representa la unificación de las técnicas anteriores. Es encargado de segmentar de manera simultánea los objetos identificables del primer plano (denominados *thing)* así como los elementos presentes en el fondo del entorno (conocidos como *stuff*), como es el caso de la vegetación, el cielo y estructuras urbanas (Miguélez, 2022). Esto se visualiza en la Figura 13(e) en la cual se observa que los objetos 'gato' son identificados y pintados de un color cada uno, pero a su vez el fondo también se identifica como otro elemento presente en la imagen.

Figura 3

Técnicas de segmentación de imágenes.

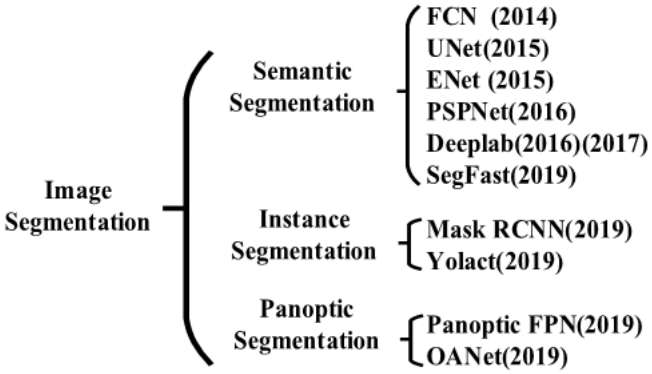


Nota. Tareas de visión por computador. a) Imagen de entrada b) Detección de objetos c) Segmentación semántica d) Segmentación de instancias e) Segmentación panóptica. Tomado de (Miguélez, 2022).

Para el presente trabajo de titulación se seleccionó la segmentación semántica debido a que el objetivo principal es identificar la presencia o ausencia de vías de acceso terrestre que sean transitables. Por lo que no se requiere contabilizar o separar calles de manera individual, sino mapear la superficie continua y transitable frente a píxeles que engloban clases tales como inundación, edificaciones, vegetación o zonas de riesgo en general.

Figura 4

Aplicación de CNN en segmentación de imágenes.



Nota. Tomado de (Z. Li et al., 2022)

Históricamente, la realización de tareas de visión por computador como la segmentación de imágenes se basaba en algoritmos tradicionales que hacían uso de descriptores manuales, filtrado de bordes y umbrales de color. El avance llega con la introducción de las redes neuronales convolucionales, mismas que permitieron generar un análisis más sólido y automatizado capaz de aprender características profundas. Esto sirvió como base para el desarrollo de arquitecturas de segmentación modernas (Z. Li et al., 2022). En la Figura 4 se visualiza algunas de las CNNs más usadas para resolver las distintas técnicas de segmentación de imágenes.

Enfocándose en la segmentación semántica, a partir de la introducción de las redes totalmente convolucionales (FCN), el campo de este tipo de segmentación experimentó un cambio de paradigma radical al permitir la clasificación de imágenes a nivel de píxel sin perder la información espacial esencial. Esta arquitectura fue la base para el posterior diseño del moderno de codificador-decodificador, el cual fue optimizado por modelos como U-Net. U-Net no solo replica la estructura simétrica para recuperar la resolución de la imagen, sino que implementa de manera estratégica un esquema de conexiones de salto y así transfiere características detalladas directamente desde las etapas de contracción a las de expansión.

Esto le permite consolidarse rápidamente como el estándar de referencia en entornos con conjuntos de datos más reducidos. (Z. Li et al., 2022)

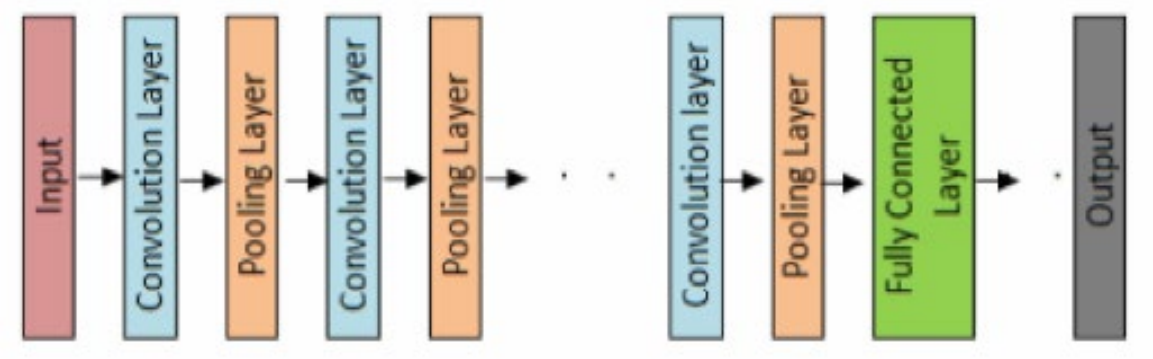
Por otro lado, se presentan propuestas más robustas como PSPNet y la serie DeepLab que permiten abordar limitaciones asociadas al contexto global y variaciones de escala en escenas que pueden resultar de mayor complejidad. En el caso de PSPNet (conocida por sus siglas en inglés como *Pyramid Scene Parsing Network*), se introdujo un módulo de agrupación piramidal que es capaz de capturar información del contexto en diferentes zonas de la imagen, mejorando así la consistencia de las predicciones. Asimismo, modelos como DeepLab marcaron un hito al explotar el uso de convoluciones dilatadas. Estas técnicas permiten que los filtros puedan ver más sin necesidad de aumentar mucho el número de parámetros ni perder detalle en las imágenes. Esto muestra cómo se busca constantemente un equilibrio entre la precisión en los detalles pequeños y la comprensión de la imagen en general (Z. Li et al., 2022).

Redes neuronales convolucionales

Las redes neuronales convolucionales (*Convolutional Neural Networks*, CNN) constituyen un tipo de red neuronal artificial diseñada para el procesamiento de información con estructura espacial, como las imágenes, a través de una arquitectura profunda formada por múltiples capas que transforman progresivamente los datos de entrada. Además, las CNN tienen la capacidad de aprender automáticamente las características representativas de los datos de entrada, y otorgan una mayor capacidad de generalización y un reconocimiento más eficiente de patrones visuales. Al mismo tiempo, durante este proceso, las primeras capas extraen características básicas como bordes, líneas y texturas, mientras que, las capas más profundas identifican patrones de mayor complejidad, en el que se incluyen formas y objetos completos, generando que este tipo de redes sea ampliamente utilizado en la clasificación, detección y segmentación de imágenes (Ghosh et al., 2019).

Figura 5

Modelo conceptual básico de una CNN



Nota. Tomado de (Ghosh et al., 2019).

Las redes neuronales convolucionales están conformadas de diferentes operaciones. Estas operaciones ayudan a sacar y procesar características de una imagen. La operación básica de estas arquitecturas es la convolución. A partir de la convolución, se crean las capas encargadas de aprender características visuales.

Convolución

Dentro del procesamiento digital de señales, la convolución es una operación lineal que combina dos funciones para generar una tercera que representa cómo la forma de una de ellas es modificada por la otra. Matemáticamente, para dos funciones continuas *f* y *g*, la convolución se define como:

$$(f * g) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau$$

Dado que las computadoras y los sistemas digitales trabajan con datos discretos, se utiliza una versión discreta en una dimensión. Esta se puede expresar con la siguiente suma:

$$(f * g)[n] = \sum_{-\infty}^{+\infty} f[m]g[n-m]$$

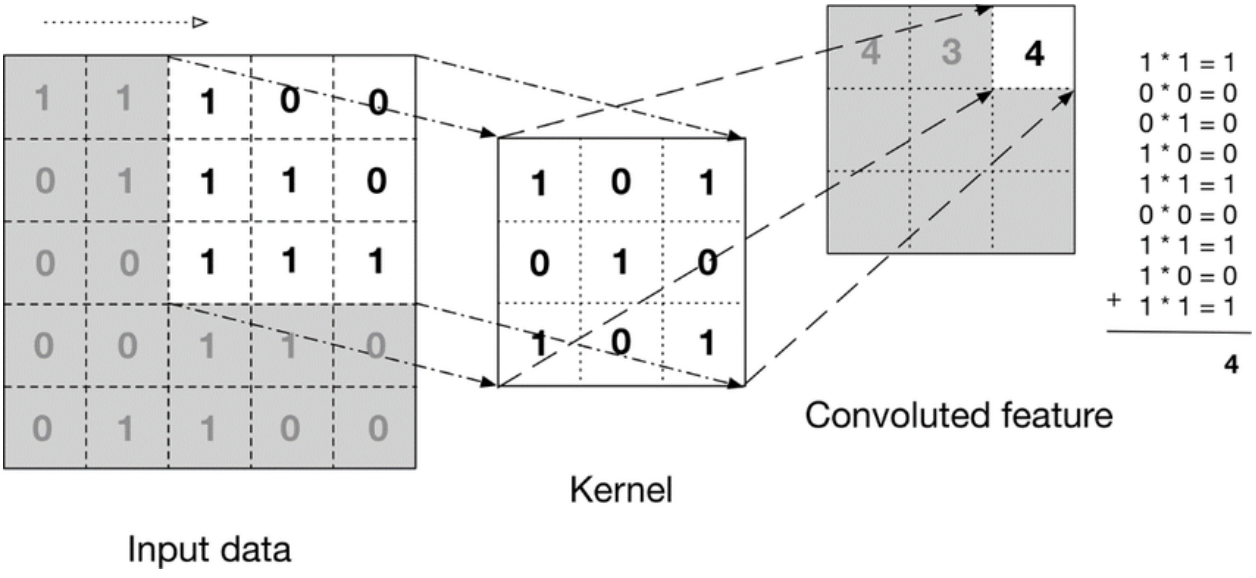
En dicha expresión matemática, la señal *g* (conocida como filtro o *kernel*) se invierte y se desplaza (con un paso o *stride*) sobre la señal de entrada *f*, de mamera que calcula la suma

de sus productos en cada punto. Cuando esta operación se traslada al procesamiento de imágenes, la ecuación se extiende a dos dimensiones para cubrir los ejes espacial y temporal (filas y columnas de píxeles) (Fronzetti, 2019).

En este sentido, la convolución es una operación matemática que se usa para extraer información relevante de una imagen mediante el desplazamiento de una pequeña matriz denominada *kernel* sobre toda la imagen de entrada. En cada posición, los valores del *kernel* se combinan con los píxeles de la región correspondiente mediante operaciones de multiplicación y suma, generando un nuevo mapa de características (*feature map*) que permite resaltar determinados patrones tal como bordes, texturas o formas, tal como se aprecia en la Figura 6. Este procedimiento es la base del aprendizaje de las redes neuronales convolucionales, ya que permite detectar automáticamente características útiles para tareas de clasificación y segmentación de imágenes (Aldea Blasco, 2019).

Figura 6

Convolución con kernel 3x3



Nota. Tomado de (Fronzetti, 2019).

A continuación, se describen los componentes esenciales para el funcionamiento interno y flujo de procesamiento de las CNN, así como una de las estrategias avanzadas que optimizan el rendimiento de las mismas.

Capas convolucionales

Las capas convolucionales constituyen el componente principal de las redes neuronales convolucionales, ya que permiten la extracción automática de las características más importantes de una imagen mediante la aplicación de filtros sobre pequeñas regiones de la misma. Al contrario de los métodos tradicionales, donde las características debían definirse manualmente, las CNN aprenden de forma automática patrones tal como bordes, texturas y formas a lo largo del proceso de entrenamiento. Al aumentar la profundidad de la red las características extraídas de las imágenes se vuelven representaciones más complejas y es así que se mejora la localización de regiones u objetos de interés. (García Villalba & Armas Vega, 2021).

Los principales parámetros de las capas convolucionales son:

Kernel

Corresponde a una matriz de pesos que va desplazándose por la imagen de entrada durante la convolución de manera que reconoce rasgos locales como bordes o texturas. Durante el entrenamiento, los valores del kernel se ajustan automáticamente para aprender las características más representativas de la imagen (Aldea Blasco, 2019).

Filtros

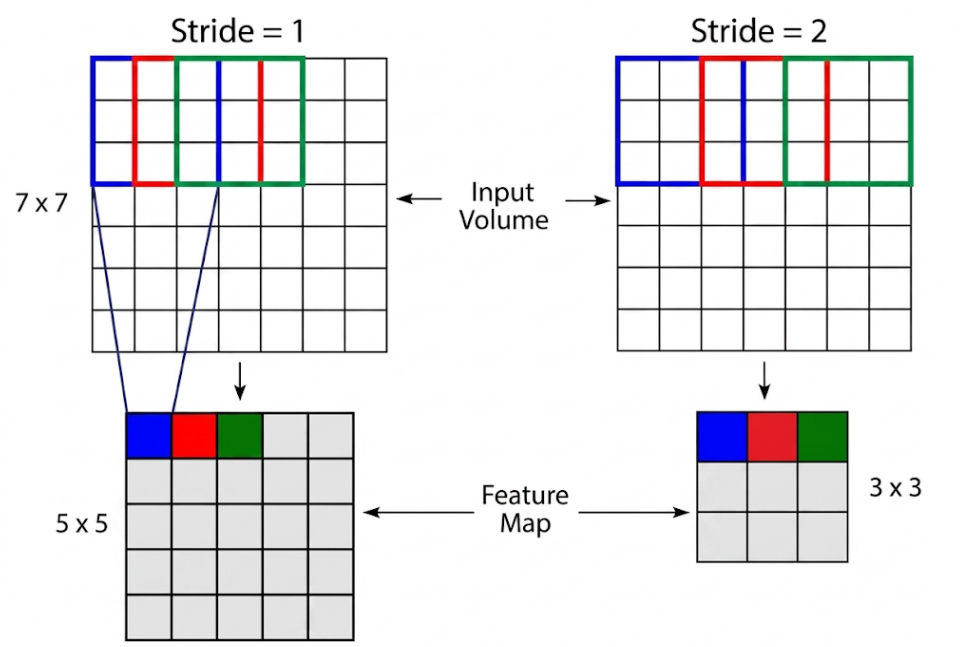
Una capa convolucional puede contener múltiples filtros, donde cada uno aprende un patrón diferente de la imagen. Como resultado, cada filtro genera un mapa de características independiente, permitiendo representar información visual cada vez más compleja conforme aumenta la profundidad de la red (García Villalba & Armas Vega, 2021).

Stride

Corresponde al número de píxeles que el kernel avanza en cada desplazamiento sobre la imagen. Un valor pequeño permite conservar mayor detalle espacial, mientras que un valor mayor reduce el tamaño de los mapas de características y disminuye el costo computacional del procesamiento (Aldea Blasco, 2019). Esto se observa en la Figura 7.

Figura 7

Stride



Nota. Comparación visual entre *stride*=1 y *stride*=2. Adaptado de (Swapna, 2020).

Funciones de activación

Las funciones de activación se aplican después de la operación de convolución con el objetivo de incorporar no linealidades en la red neuronal, permitiendo así que la red neuronal aprenda relaciones y patrones más complejos presentes en las imágenes. Cada neurona procesa la información recibida y, mediante una función de activación, determina cómo será transmitida a la siguiente capa. Como resultado, los mapas de características obtenidos tras la convolución pueden representar información de mayor utilidad para el proceso de aprendizaje y

mejorar la capacidad de la red para resolver tareas de visión por computador (Aldea Blasco, 2019).

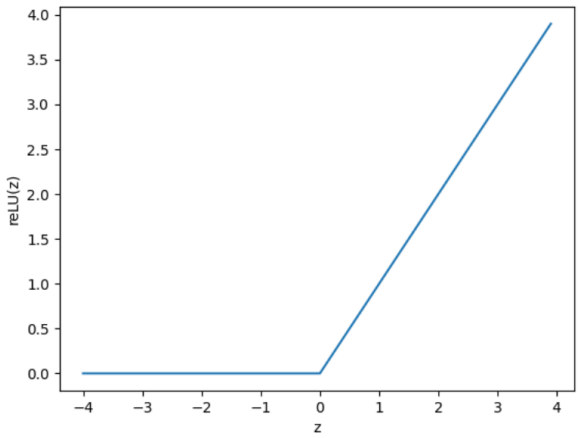
A continuación, se describen las funciones de activación empleadas en las arquitecturas desarrolladas en esta investigación.

ReLU (*Rectified Linear Unit*)

La función ReLU es muy usada en redes neuronales convolucionales debido a su simplicidad y eficiencia computacional. Funciona de manera que conserva los valores positivos de entrada y reemplaza por cero aquellos valores negativos, lo que facilita el entrenamiento de redes profundas ya que acelera la convergencia y reduce problemas asociados al desvanecimiento del gradiente (García Villalba & Armas Vega, 2021). Debido a esto, ReLU se emplea de manera generalizada en las capas ocultas de las arquitecturas de segmentación implementadas. En la Figura 8 se visualiza el comportamiento de esta función.

Figura 8

Gráfica de la función ReLU



Nota. Tomado de (García Villalba & Armas Vega, 2021).

La función de activación ReLU se expresa de la siguiente manera:

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

Sigmoid

La función Sigmoid transforma los valores de entrada en un rango comprendido entre 0 y 1, de forma que permite interpretar la salida de la red como una probabilidad. Debido a esta característica, es ampliamente utilizada en problemas de clasificación y segmentación binaria (García Villalba & Armas Vega, 2021). Su expresión matemática es:

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

En el presente trabajo, la función Sigmoid se empleó en la capa de salida de los modelos para estimar la probabilidad de que cada píxel pertenezca a la clase "*Road Non-Flooded*", facilitando la posterior generación de la máscara binaria correspondiente.

Batch Normalization

La *Batch Normalization* es una técnica que se usa en redes neuronales convolucionales para normalizar las activaciones que se producen durante el entrenamiento. Se aplica entre la capa convolucional y la función de activación con el objetivo de mantener una distribución más estable de los datos que circulan por la red. Esto permite que se reduzca el desplazamiento de covariables internas, mejore la estabilidad del aprendizaje y emplea tasas de aprendizaje más altas, disminuyendo el tiempo para entrenar el modelo para así favorecer su rendimiento (Swapna, 2020).

Pooling

Las capas de *Pooling* realizan un proceso de reducción espacial también conocido como *downsampling* sobre los mapas de características obtenidos después de las operaciones de convolución y activación, explicados previamente. El objetivo principal es la disminución de las dimensiones de estos mapas, reduciendo la cantidad de parámetros y el costo computacional del modelo, sin perder la información más representativa. Además, esta operación ayuda a que la red sea mejor en generalizar y hace que sea menos probable que se sobreajuste durante el entrenamiento. (Swapna, 2020).

A continuación, se describen las operaciones de *pooling* más empleadas.

Max Pooling

Esta operación consiste en dividir el mapa de características en pequeñas regiones y conservar únicamente el valor máximo de cada una de ellas. De esta manera, se preservan las características más relevantes detectadas por la red, como bordes o patrones destacados, al mismo tiempo que se reduce la resolución espacial del mapa de características. Por tanto, la operación de *pooling* más utilizada en redes neuronales convolucionales es esta porque conserva la información más importante. (Swapna, 2020).

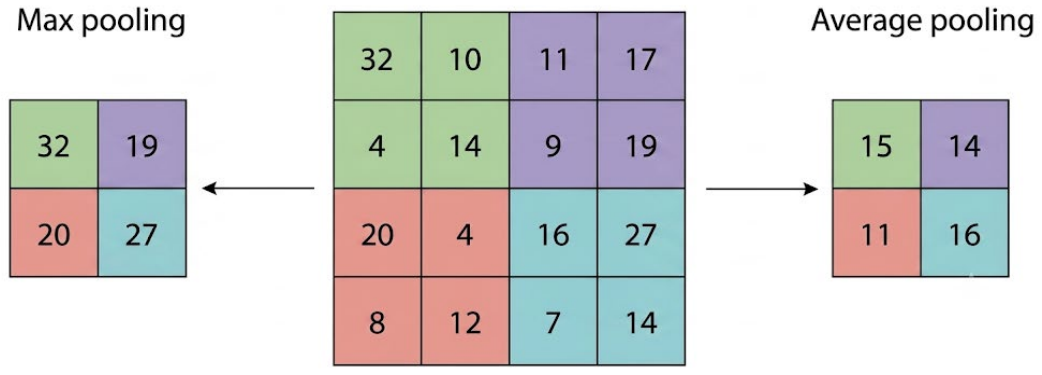
Average Pooling

Esta operación calcula el valor promedio de los elementos contenidos en cada región del mapa de características, generando una representación más suavizada de la información. A pesar que esta operación conserva menos detalles locales a diferencia de *Max Pooling*, es muy eficaz para capturar información contextual de la imagen (Swapna, 2020).

En la Figura 9 se puede visualizar cómo funcionan ambas capas de *pooling* previamente mencionadas.

Figura 9

Capas de Pooling



Nota. Adaptado de (Swapna, 2020)

Encoder y Decoder

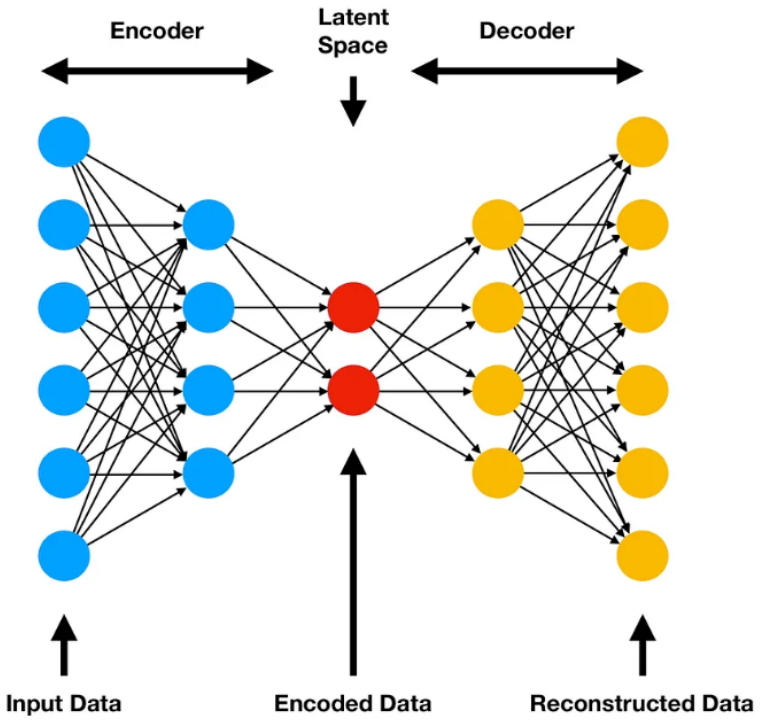
Las arquitecturas *Encoder–Decoder* son ampliamente utilizadas en tareas de segmentación semántica debido a su capacidad para extraer y reconstruir información espacial de una imagen. El *encoder* o codificador obtiene características importantes de la imagen de entrada mediante capas convolucionales y operaciones de reducción espacial como el *pooling*. Es así que la resolución espacial de la imagen disminuye progresivamente a medida que incrementa el número de características aprendidas por la red al ir extrayendo cada vez patrones más complejos que pasan a generar una representación comprimida también conocida como espacio latente (Kainat, 2023).

Luego de ese comprimido interviene el decodificador para reconstruir progresivamente la información espacial hasta igualar la resolución con la que ingresó la imagen a la red. Para hacer esto, utiliza operaciones especiales que aumentan la cantidad de detalles en los mapas de características. La calidad del resultado final depende de la capacidad del codificador para obtener la información importante y del decodificador para recuperar los detalles espaciales necesarios durante el proceso de reconstrucción (Kainat, 2023).

La Figura 10 presenta el esquema general de una arquitectura Encoder–Decoder empleada en redes neuronales convolucionales, en la cual se puede observar el flujo de datos de entrada, que pasa a convertirse en información codificada para luego realizar su respectiva decodificación y dar como resultado final las imágenes reconstruidas de todo este proceso de entrenamiento con redes neuronales convolucionales.

Figura 10

Arquitectura Encoder – Decoder.



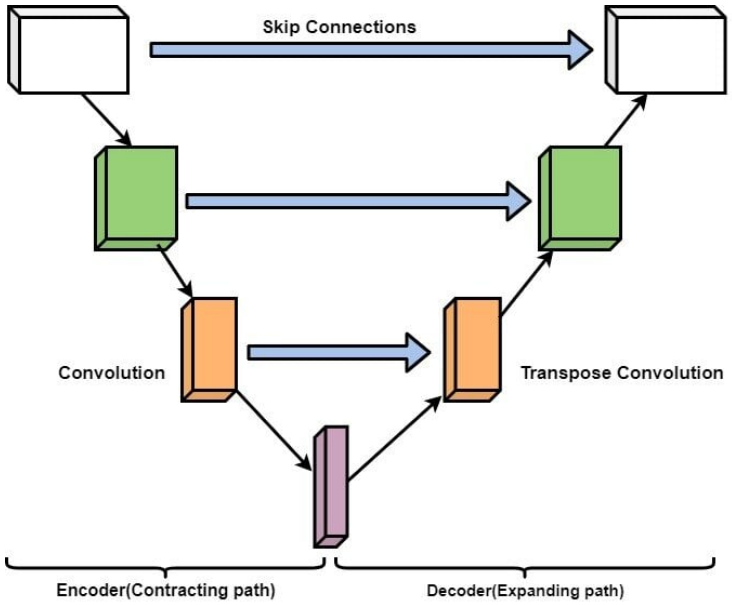
Nota. Tomado de (Kainat, 2023).

Skip Connections

Son conexiones que permiten transmitir información desde capas anteriores hacia capas más profundas de la red, evitando que parte de las características extraídas durante las primeras etapas del procesamiento se pierdan. Esta estrategia ayuda a que la información y el gradiente fluyan mejor cuando se entrena. Esto permite construir redes más profundas y mejorar la calidad de las características que se aprenden. En arquitecturas de segmentación, estas conexiones permiten combinar información de diferentes niveles entre el *encoder* y el *decoder*. Esto ayuda a recuperar detalles espaciales y a obtener resultados de segmentación más precisos (Ji et al., 2021). La Figura 11 muestra el principio general de funcionamiento de las Skip Connections.

Figura 11

Diagrama con skip connection



Nota. Tomado de (Lata, 2022).

Backbones preentrenados

El aprendizaje por transferencia es una estrategia que permite reutilizar el conocimiento adquirido por un modelo previamente entrenado para resolver una nueva tarea. Su principal ventaja es que aprovecha lo que ha aprendido de grandes conjuntos de datos. De esta manera, no es necesario entrenar una red desde cero. Esto ayuda a que un entrenamiento de red neuronal converja más rápido, sobre todo cuando solo se dispone de unas pocas imágenes para el nuevo problema. (Pan & Yang, 2009). En este contexto, el backbone constituye la estructura principal de una red neuronal convolucional y actúa como el encargado de extraer las características de la imagen de entrada mediante múltiples capas convolucionales, generando representaciones visuales cada vez más complejas (Matamoros Pinto, 2025).

Con frecuencia, los *backbones preentrenados* provienen de repositorios masivos y entre ellos *ImageNet* destaca como la referencia principal en visión por computador. Este conjunto de datos está organizado de acuerdo con la jerarquía semántica de *WordNet* que comprende

millones de imágenes repartidas en miles de clases que enseñan a la red a reconocer rasgos genéricos del entorno visual. Luego, estas características se pueden reutilizar en tareas de segmentación de imágenes. Por eso, *ImageNet* es uno de los principales referentes para el preentrenamiento de modelos de redes neuronales convolucionales (Deng et al., 2021).

Capítulo III: Desarrollo del sistema propuesto

En este capítulo se describe la metodología empleada para el desarrollo del presente trabajo de integración curricular. Se detalla el flujo de trabajo, las herramientas utilizadas y las configuraciones necesarias para el entrenamiento de los tres modelos a ser evaluados. La secuencia presentada a continuación permitirá comprender la estructura del trabajo desde la preparación de los datos hasta la inferencia de los modelos de redes neuronales y los análisis de los resultados.

Es importante señalar que la principal contribución de este trabajo no consiste únicamente en la comparación del desempeño de diferentes arquitecturas de segmentación semántica. Como parte de la investigación se desarrolló un flujo de trabajo completo que comprendió la construcción de un conjunto de datos binario orientado a la identificación de vías de acceso terrestre, la corrección y validación manual de las máscaras, la generación de pseudoetiquetas para ampliar el conjunto de entrenamiento, el preprocesamiento uniforme de las imágenes y el establecimiento de un protocolo reproducible de entrenamiento y evaluación. Este procedimiento permitió obtener un sistema de segmentación preparado para ser utilizado y evaluado bajo las mismas condiciones experimentales. Este proceso de detalla a lo largo de este capítulo.

Entorno de desarrollo

Para el desarrollo del trabajo se implementaron diversas herramientas de acceso libre, lo que facilitó el almacenamiento de datos, entrenamiento de modelos y visualización de los resultados obtenidos. A continuación, se detallan los componentes técnicos que integran el entorno de desarrollo.

Almacenamiento y gestión de datos: Google Drive

Esta plataforma se usó como el sistema de almacenamiento en la nube para organizar el conjunto de datos extraído de FloodNet, es decir, las imágenes aéreas en escenarios de inundación con sus respectivas máscaras obtenidas a partir del dataset mencionado, así como

las máscaras generadas de forma manual. Además, en este entorno se guardaron los archivos correspondientes a la preparación de los datos, el entrenamiento y los modelos obtenidos durante el proceso. Su vinculación directa con el entorno de ejecución (Google Colab) fue de gran ayuda para acceder de manera directa a la información almacenada sin requerir de copias locales de los archivos.

Herramienta para la edición de anotaciones: MakeSense.ai

Según Skalski (2019) MakeSense.ai es una herramienta online de código abierto especializada para el etiquetado de imágenes en proyectos de visión por computador, ya que optimiza la preparación de datasets gracias a los módulos de reconocimiento de imágenes y detección de objetos que posee dentro de su interfaz gráfica basada en navegador, es decir, no requiere de una instalación de software en el computador.

Para este trabajo, la herramienta fue empleada para la revisión y elaboración manual de máscaras binarias de segmentación mediante el módulo de detección de objetos a fin de identificar visualmente e ir trazando mediante polígonos los segmentos correspondientes a vías no inundadas de las imágenes aéreas. Esta elaboración manual de las máscaras fue necesaria para cierta cantidad de imágenes ya que no todo el dataset de FloodNet contenía anotaciones correctas o en su defecto estaban vacías. Una vez concluido el etiquetado manual se exportan los archivos con las anotaciones geométricas realizadas en formato COCO JSON desde la plataforma. Es así que los datos quedan preparados para incorporarse en el dataset general con el fin de procesarlos posteriormente para el entrenamiento.

Plataforma de Desarrollo: Google Colab

Google Colab también conocido como Google Colaboratory es un entorno de desarrollo basado en la nube, el cual permite la creación y ejecución de cuadernos interactivos de Jupyter. Colab facilita la programación tanto en Python como en R, así como otorga accesos a recursos de cómputo optimizados (Kimm et al., 2021). Para el presente trabajo se usó la versión gratuita de la plataforma. Al revisar los recursos del entorno de ejecución real, se

verificó que el sistema asignó un perfil de memoria estándar contando con una capacidad máxima de 12.7 GB de memoria RAM de sistema, 15.0 GB de VRAM de la GPU y 112.6 GB de almacenamiento en disco.

En proyectos de aprendizaje automático, Google Colab destaca por permitir el cambio de un entorno basado en CPU a uno potenciado por aceleradores de hardware. Esta capacidad resulta fundamental al procesar operaciones matriciales intensivas en redes profundas. La plataforma en su versión gratuita da acceso a GPU (Unidades de Procesamiento Gráfico) y TPU (Unidades de Procesamiento de Tensores): mientras la primera ofrece gran versatilidad en cómputo paralelo, la TPU optimiza drásticamente los tiempos en tareas masivas de aprendizaje profundo (Kimm et al., 2021).

Al momento de seleccionar el acelerador a usar, en primera instancia se pretendía usar una TPU, sin embargo, existe una incompatibilidad del entorno de software ya que al indagar sobre las versiones del entorno de ejecución se identificó que la librería a usarse para el entrenamiento (TensorFlow) no se encuentra incluida en los entornos de ejecución de TPU. Además, existiría una subutilización del acelerador debido a que la arquitectura para la que fue diseñada la TPU implica procesar volúmenes masivos de datos simultáneamente. Debido a que el presente trabajo no cuenta con una gran magnitud de información, el sistema sería ineficiente al desperdiciar su capacidad de gestión interna y desgastar recursos, por lo tanto, en el desarrollo del trabajo se seleccionó una aceleración basada en la GPU NVIDIA Tesla T4.

Asimismo, se aprovechó la integración de funciones de asistencia con inteligencia artificial presentes en el entorno de desarrollo con el fin de agilizar etapas de automatización de código y depuración del mismo. La plataforma de Colab integra el modelo Gemini 2.5 Flash, mismo que se usó como apoyo para la extracción del conjunto de datos de FloodNet del archivo con formato “.tar”, así como para la conversión de archivos JSON obtenidos de Makesense.ai hacia imágenes png correspondientes a las máscaras binarias que serán posteriormente usadas para el entrenamiento.

Bibliotecas

Para el desarrollo del proyecto se utilizaron diversas bibliotecas de Python enfocadas en tareas aprendizaje profundo, procesamiento de datos y visualización de resultados obtenidos. En la Tabla 1 se contempla la función que cumplió cada una de las bibliotecas dentro del trabajo.

Tabla 1

Bibliotecas usadas dentro de entorno de desarrollo

Biblioteca	Descripción	Referencia
TensorFlow / Keras	Empleada para la construcción, entrenamiento y evaluación de las tres arquitecturas de segmentación semántica basadas en CNN.	(Abadi et al., 2016)
NumPy	Usada para el manejo de arreglos multidimensionales y almacenamiento del conjunto de datos en formato .npy permitiendo una carga rápida de los datos previo al entrenamiento de cada modelo.	(Harris et al., 2020)
OpenCV (cv2)	Adquisición y preprocesamiento de dataset.	(Bradski, 2000)
Matplotlib	Visualización de imágenes, máscaras predichas, curvas de entrenamiento y gráficos estadísticos de los resultados.	(Hunter, 2007)
Pillow (PIL)	Lectura, creación y almacenamiento de imágenes durante la generación de máscaras que fueron etiquetadas manualmente.	(Clark, 2024)
json	Lectura y procesamiento de los archivos de anotaciones del conjunto de datos FloodNet, así como de los archivos COCO JSON generados por Makensense.ai	(Python Software Foundation., 2025a)
os	Gestión de directorios, creación de carpetas y manipulación de rutas de archivos. Mediante esta librería se realizó la división del conjunto de datos general en tres carpetas correspondientes a entrenamiento, validación y prueba.	(Python Software Foundation., 2025b)
tarfile	Usado para realizar la extracción de los datos que se encontraban en el archivo comprimido .tar correspondiente al conjunto de datos de FloodNet.	(Python Software Foundation., 2025c)

supervisely	Descarga y gestión del conjunto de datos comprimido de FloodNet que se encontraba en la plataforma Supervisely.	(Supervisely., 2024)

Nota. Elaboración propia.

Base de datos

Para realizar el entrenamiento de los modelos de redes neuronales convolucionales aplicados a la segmentación semántica se hizo uso del dataset de acceso libre FloodNet. Este conjunto de datos se empleó específicamente para la evaluación de los daños post – inundación luego del paso de huracán Harvey cerca de Texas y Luisiana en agosto de 2017. Las imágenes aéreas fueron capturadas mediante vehículos aéreos no tripulados, específicamente con un cuadricóptero DJI Mavic Pro. (Rahnemoonfar et al., 2020)

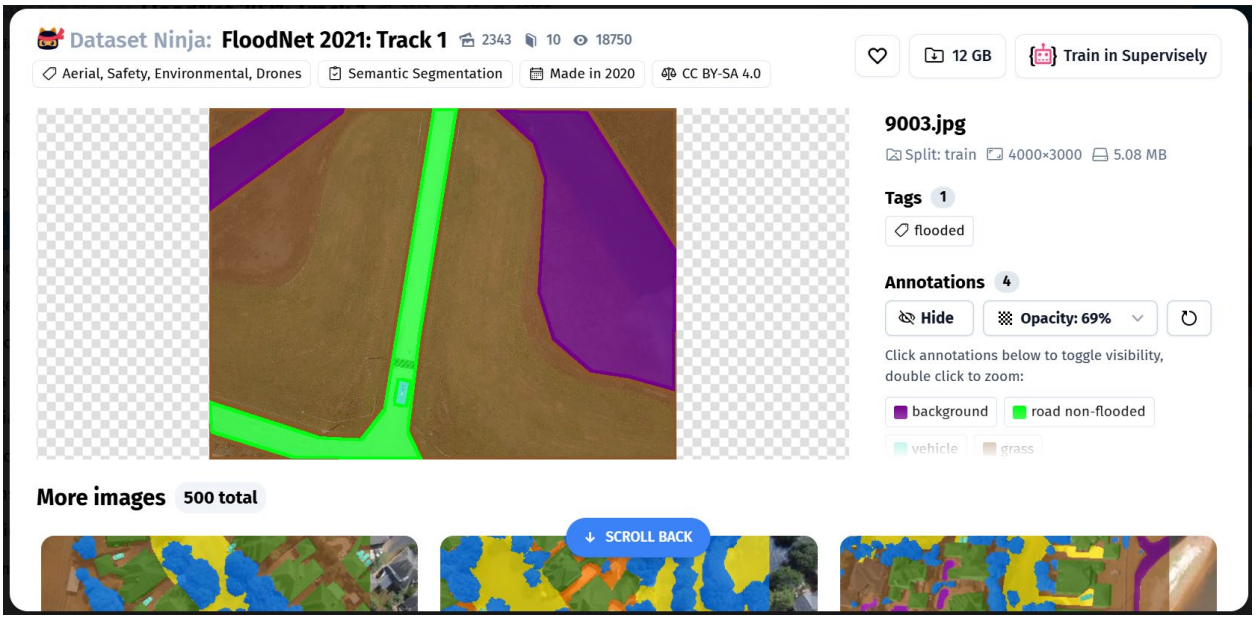
El dataset FloodNet se encuentra alojado y es distribuido oficialmente a través de la plataforma de desarrollo de tareas de visión por computador llamada Supervisely. Fue imperativo descargar de este entorno porque ahí fueron realizadas las tareas iniciales de anotación, etiquetado y estructuración de las imágenes, de forma que los creadores del dataset dividieron el conjunto de datos completo de 2343 imágenes en tres carpetas principales correspondientes a entrenamiento (*train*), que concentra el 60% de las muestras, mientras que validación (*validation*) y prueba (*test*) recibieron un 20% del volumen total cada una (Rahnemoonfar et al., 2020). Asimismo, acorde al número total de imágenes, se presentó en el dataset los archivos JSON correspondientes a las anotaciones. Sin embargo, es importante recalcar que solo una parte de las imágenes de entrenamiento tenía correctamente etiquetadas sus clases, mientras que las imágenes correspondientes a las otras carpetas no estaban etiquetadas en su totalidad, es decir, los archivos JSON estaban vacíos.

Las imágenes aéreas almacenadas en el dataset presentan una alta resolución de 4000x3000 píxeles. Por otro lado, los JSON contienen anotaciones detalladas para la identificación de 10 clases de objetos en las imágenes, tales como césped, árbol, carretera

inundada, carretera no inundada, edificio inundado, edificio no inundado, agua, vehículo, piscina y fondo. En la Figura 12 se visualiza un ejemplo de las anotaciones presentes en las imágenes.

Figura 12

Anotaciones de clases identificadas en imágenes de conjunto de datos de FloodNet.



Nota. Tomado de (Rahnemoonfar et al., 2020)

Construcción del conjunto de datos

A continuación, se explica el proceso realizado para transformar los datos crudos de FloodNet en el dataset final optimizado a utilizarse para entrenar los modelos de CNN para la segmentación binaria de las imágenes.

Selección de la clase objetivo

Una vez analizadas todas las clases identificadas en FloodNet, se aisló exclusivamente la categoría de carreteras no inundadas (*Road Non-Flooded)*, entendiendo el contexto para el cual se requiere el desarrollo del presente módulo de software en donde se busca identificar vías de acceso potencialmente transitables para vehículos de emergencia. Por lo que el resto de clases fueron descartadas y agrupadas bajo la etiqueta de zonas no accesibles o no

transitables, de manera que finalmente se manejan dos tipos de clases orientando el problema a la segmentación binaria.

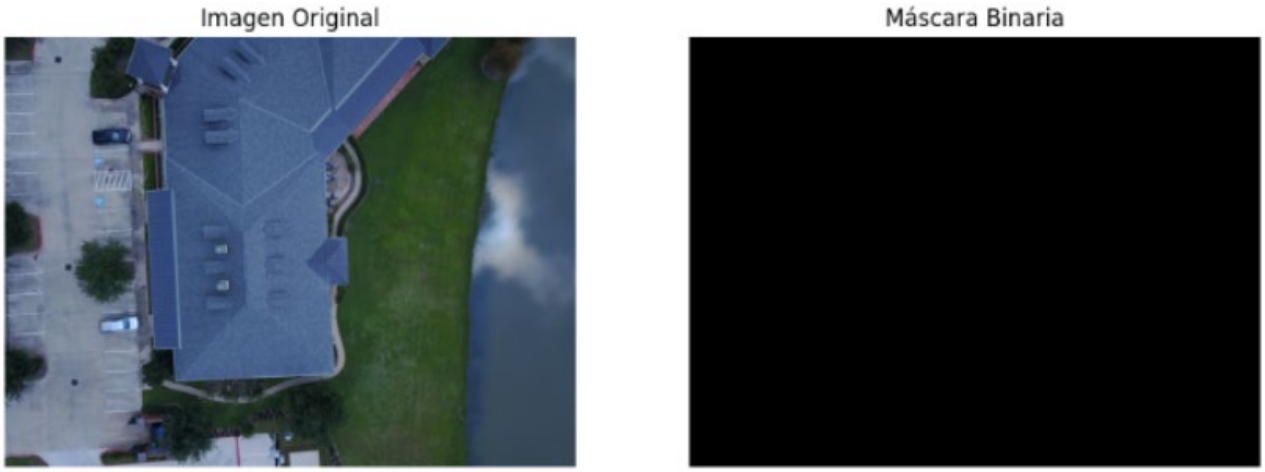
Generación automática de máscaras

Al extraer las imágenes con sus respectivas anotaciones del dataset original, se evidenció que las etiquetas se encontraban estructuradas en archivos con formato JSON como ya antes se había mencionado. Es así que la información original contenía la geometría de las clases representada en polígonos vectoriales y dado que las redes de segmentación requieren mapas de referencia (conocidos como *ground truth*) en formato matricial, fue indispensable transformar esos polígonos a imágenes de máscaras binarias. Este procedimiento se automatizó con scripts de Python (apoyados en OpenCV y NumPy dentro de Google Colab) los cuales leyeron los archivos JSON de *Supervisely* y guardaron las máscaras finales en formato PNG.

Aunque el volumen inicial del dataset era de 2343 imágenes se hallaron limitaciones críticas en las etiquetas originales. Se detectó que la gran mayoría de los archivos no contenían anotaciones asociadas a la clase de interés, que para el presente trabajo es ´carreteras no inundadas´. Asimismo, se constató que las carpetas destinadas a validación (*validation*) y prueba (*test*) presentaban archivos JSON completamente vacíos, tal como lo reportaba el autor del dataset. Para comprobar esta condición de manera empírica, se ejecutó una primera prueba generando la máscara binaria de una imagen que visualmente mostraba un camino transitable; el resultado fue una imagen completamente negra debido a la ausencia absoluta de metadatos de segmentación para esa vía tal como se aprecia en la Figura 13.

Figura 13

Generación máscara binaria a partir de JSON de dataset

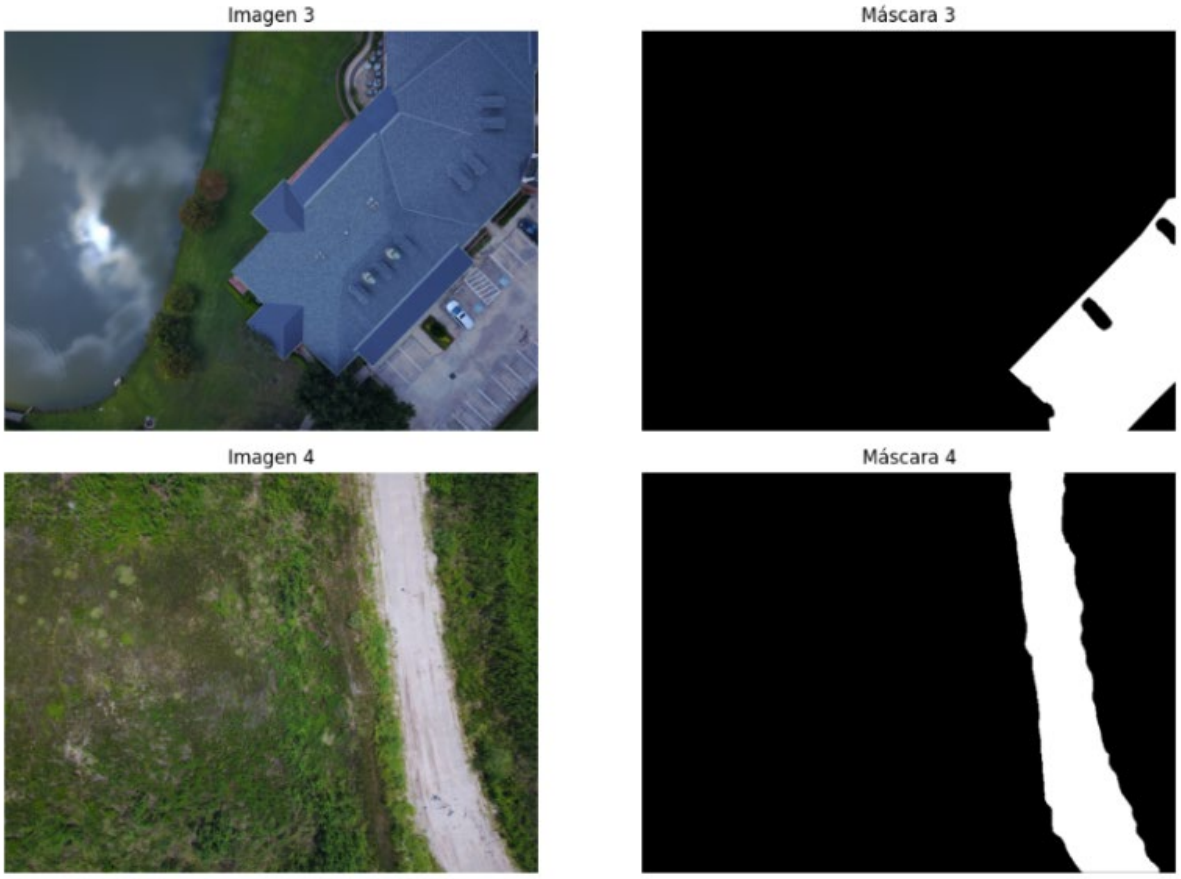


Nota. El gráfico representa la generación de una máscara binaria correspondiente al mapa de referencia con el que se entrenarán los modelos. Fue tomada de la carpeta *Test* del dataset original para verificar la inexistencia de anotaciones en las imágenes.

En contraste, para comprobar el desempeño del *script* diseñado para la elaboración de las máscaras binarias, se ejecutó el mismo sobre archivos presentes en la carpeta *Train* sobre archivos JSON que si disponían de anotaciones en su contenido. Como se observa en la Figura 14 esta comprobación rápida permitió visualizar tanto la imagen original como la generación exitosa de sus respectivas máscaras binarias a partir de los vectores del JSON, corroborando que el código interpretaba correctamente las coordenadas de la clase requerida.

Figura 14

Verificación generación máscara binaria a partir de JSON de dataset



Nota. El gráfico de la generación de máscaras binarias permite validar la efectividad del *script* empleado para identificar la clase de interés y generar el mapa binario resultante como una imagen PNG.

Por otra parte, para corregir la falta de etiquetas se implementó un script de filtrado para seleccionar únicamente las imágenes cuyos archivos JSON contenían anotaciones válidas de la clase objetivo de caminos no inundados. Luego de la revisión se identificó que, de las 1445 capturas iniciales de la carpeta de entrenamiento, apenas 194 contaban con la etiqueta de carreteras no inundadas. Este desbalance de clases hizo necesaria una reestructuración del almacenamiento en Google Drive antes de continuar con la depuración del conjunto de datos.

Con el objetivo de proveer al modelo casos de control y mejorar su capacidad de discriminación visual, se resolvió estructurar el conjunto inicial con las 194 imágenes positivas y se incorporaron 204 imágenes de casos completamente negativos (imágenes que contenían lodo, agua o vegetación, pero ninguna calle transitable, generando máscaras totalmente negras). Es importante señalar que la incorporación limitada de estas muestras negativas cumple un rol clave en el entrenamiento, ya que enseña al modelo a prevenir falsos positivos en escenarios complejos donde el entorno domina sobre los píxeles de la carretera.

Dado que un volumen total de 398 imágenes resultaba insuficiente para garantizar la solidez del modelo ante la variedad de terrenos y escenarios tras un desastre de inundación, se procedió a realizar una inspección visual para la selección de nuevas muestras y así fortalecer el conjunto de datos a usarse dentro del trabajo de integración curricular. Las imágenes sobrantes del dataset original pasaron a una fase de selección previa, para luego proceder a su etiquetado manual y, más adelante, aplicar una técnica de pseudoetiquetado mediante un modelo convolucional auxiliar basado en la arquitectura U-Net para fortalecer el dataset. De acuerdo con los principios establecidos por Ronneberger et al. (2015) en el desarrollo de la arquitectura U-Net, si bien este modelo destaca por su alta eficiencia en el uso de datos, el entrenamiento con un número reducido de muestras base incrementa el riesgo de sobreajuste (*overfitting*) y limita la capacidad de la red para reconocer variaciones morfológicas complejas. Por ese motivo, en la literatura se menciona que para consolidar un modelo sólido y con alta capacidad de generalización en tareas de segmentación, es fundamental expandir el volumen del dataset hacia un umbral técnico óptimo.

Etiquetado manual mediante MakeSense.ai

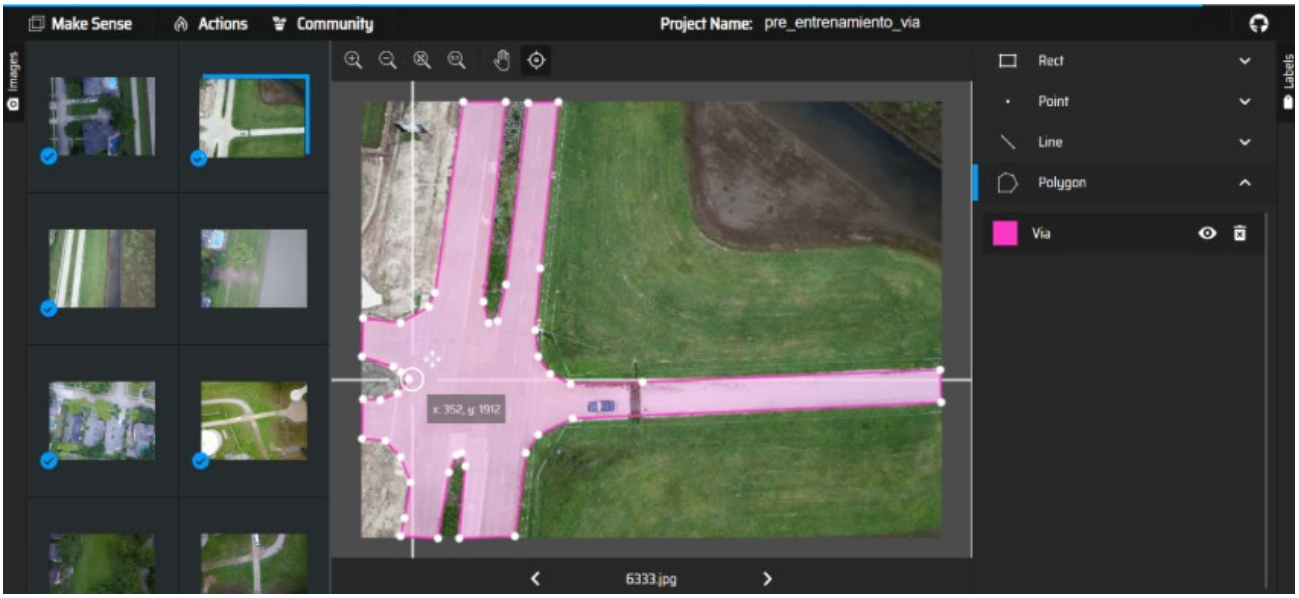
Para mitigar el desbalance de clases identificado con anterioridad, se realizó una curación y etiquetado manual de las imágenes remanentes del dataset que efectivamente presentaban caminos transitables, pero no tenían archivos JSON con sus anotaciones. Este

proceso de etiquetado manual fue necesario debido a la importancia de contar con suficientes patrones de calidad antes de recurrir a la fase de *pseudolabelling*.

La delimitación de la clase de interés se realizó mediante la herramienta de anotación por polígonos Makesense.ai. Esta herramienta permitió trazar puntos consecutivos sobre los contornos exactos de las carreteras que no tenían índices de inundación, permitiendo determinar geometrías irregulares, curvas pronunciadas y obstrucciones que suelen verse en caminos luego de desastres relacionados a inundaciones. En la Figura 15 se visualiza el entorno de trabajo sobre el cual se realizaron las anotaciones, en donde la etiqueta ´Via´ corresponde a la clase de carretera no inundada. Además, se observa la precisión con la que fueron delineados los límites del camino transitable, asegurando que el resto de elementos circundantes queden excluidos del mapa objetivo.

Figura 15

Etiquetado manual en Makesense.ai



Nota. Visualización de las anotaciones realizadas al conjunto de imágenes seleccionadas visualmente para aportar a la solidez del dataset para el entrenamiento de arquitecturas.

Una vez completadas las anotaciones sobre las imágenes seleccionadas, se exportaron las etiquetas en formato COCO JSON (*Common Objects in Context*). Esta estructura es muy útil gracias a su amplia estandarización en entornos de *Deep Learning* debido a la organización eficiente de metadatos de las imágenes, identificadores de categorías y arreglos de coordenadas (x, y) de cada polígono dentro de un único archivo indexado. (Lin et al., 2014)

Generación de pseudoetiquetas

El proceso de segmentar y delimitar elementos de infraestructura mediante técnicas de aprendizaje profundo generalmente necesita mucha información que ha sido revisada y etiquetada a mano, lo que implica un gran desgaste temporal. Para solucionar este problema, se decidió usar una estrategia de aprendizaje semi-supervisado en inteligencia artificial que se basa en la generación de pseudoetiquetas. Como sintetizan Kage & Rothenberger (2025), esta técnica consiste en entrenar un modelo inicial o auxiliar utilizando un conjunto reducido de datos meticulosamente etiquetados y usar ese modelo para posteriormente predecir etiquetas en datos que no han sido etiquetados, lo que permite expandir significativamente la muestra estadística reduciendo el desgaste en tareas repetitivas como lo es la realización de etiquetas o anotaciones en imágenes a mano.

A continuación, se detalla la estructura del modelo auxiliar implementado específicamente para cumplir con esta tarea de pre-etiquetado.

Modelo auxiliar

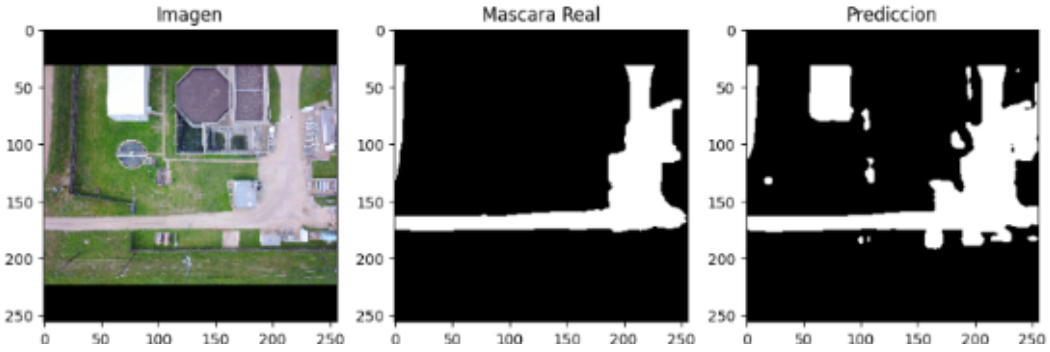
Para la generación de pseudoetiquetas se implementó una arquitectura U-Net configurada para un tamaño de entrada de 256x256x3 píxeles. Para la construcción de este modelo, se estructuró el codificador a partir de bloques convolucionales bidimensionales progresivos, iniciando con un filtro de 32, mismo que se duplica sucesivamente en cada capa hasta llegar a un *bridge* de 512 filtros en donde se aplica la técnica de regularización que apaga aleatoriamente neuronas y sus conexiones en el entrenamiento conocida como *Dropout* con un valor de 0.3. Luego el decodificador va recuperando la resolución, es decir, las dimensiones

espaciales mediante la operación de sobremuestreo conocida como *Upsampling* en combinación con conexiones de salto hasta finalizar con una activación *Sigmoid* para generar la máscara binaria.

El primer ciclo de optimización se ejecutó utilizando el optimizador Adam, una función de pérdida basada en Dice Loss y un límite de 50 épocas con un tamaño de lote (*batch size*) de 8 muestras. Adicional, se implementaron mecanismos de monitoreo como el *early stopping* con una paciencia de 10 épocas y *ReduceLROnPlateau* con un factor de reducción de 0.5 y paciencia de 5 épocas. Para el entrenamiento de prueba inicial se hizo uso de las 398 imágenes ya etiquetadas en el dataset original, debido a que sus máscaras binarias ya fueron generadas automáticamente. Sin embargo, como se aprecia en la Figura 16, al realizar la inspección visual de los resultados se identificó que el modelo predecía correctamente la geometría alargada y continua de las carreteras, sin embargo, tenía errores de contexto visual debido a la variabilidad de datos limitada por la insuficiente cantidad de muestras etiquetadas del dataset.

Figura 16

Predicción realizada de primer entrenamiento de modelo auxiliar.



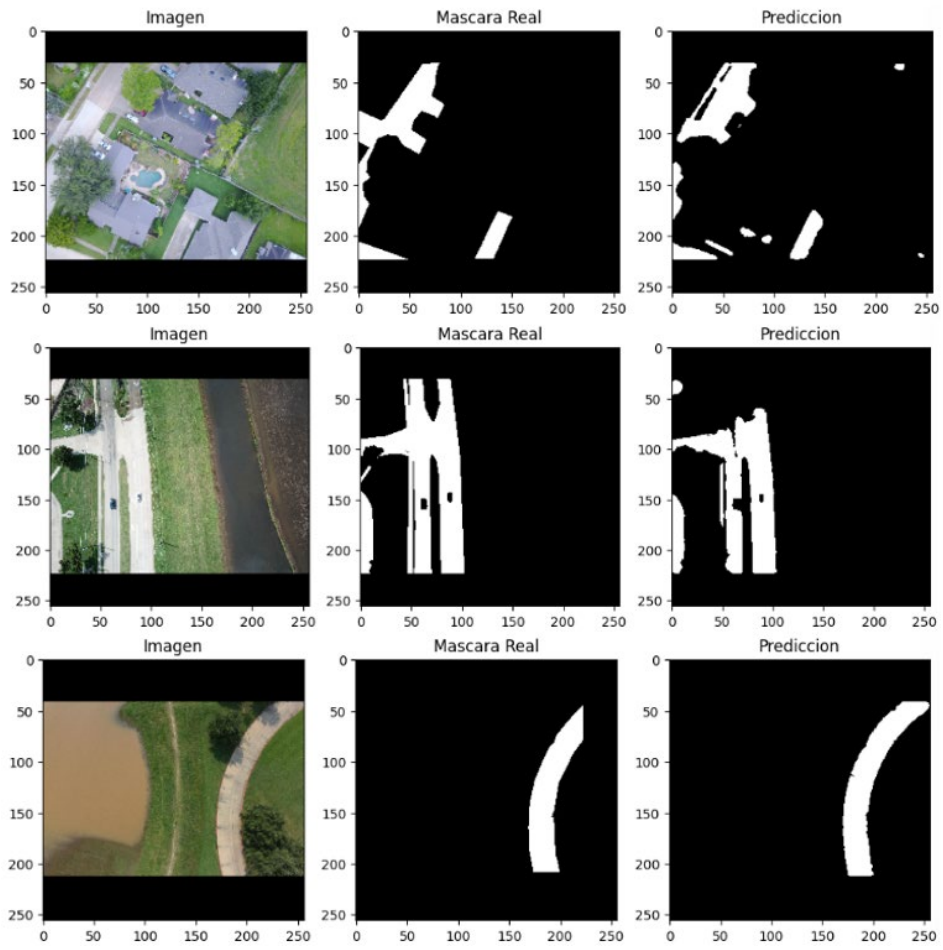
Nota. Elaboración propia

Para mejorar las capacidades de segmentación y reducir la necesidad de extensas correcciones manuales detalladas en las máscaras predichas por la primera versión del modelo, se amplió el conjunto inicial integrando 300 imágenes con sus mascaras elaboradas

mediante etiquetado manual. Al recargar los pesos preentrenados y reanudar el proceso con un volumen de datos enriquecido, el modelo mejoró significativamente su respuesta en situaciones complejas, de forma que las máscaras binarias que predijo este modelo mejorado se ajustaron más a la realidad, capturando en ciertas ocasiones detalles estructurales más precisos que las máscaras de referencia iniciales, tal como se visualiza en la Figura 17. Es así que, una vez determinada la precisión del modelo de un 98.39% y validado gracias a inspección visual, se procedió a entrenar el modelo con 500 imágenes restantes para así concluir con la generación de las pseudoetiquetas.

Figura 17

Predicciones realizadas con modelo auxiliar mejorado



Nota. Elaboración propia.

Revisión de máscaras generadas por modelo auxiliar

Aunque se depuró el modelo auxiliar destinado a la generación de pseudoetiquetas para reducir errores, la literatura advierte que puede ocurrir inconvenientes de sesgo de confirmación si se llegan a utilizar etiquetas correctas en el entrenamiento principal (Kage & Rothenberger, 2025). Eso ocasionaría que se cometan más errores si se introducen datos inadecuados en el dataset final para el entrenamiento de las 3 arquitecturas. Es por esto que para evitar errores se agregó una etapa de revisión humana. Para esta fase, en primer lugar, se creó una carpeta dentro de Google Drive llamada Dataset Auxiliar, en donde se guardaban las máscaras generadas por el modelo auxiliar para realizar una inspección visual de las mismas previo a introducirlas al Dataset final. Asimismo, se realizaron correcciones manuales en los casos que se requería eliminar errores para así asegurar una delimitación precisa en cuanto a las vías en estado de no inundación. De las 500 imágenes, 307 fueron seleccionadas para ingresar al Dataset Final.

Dataset Final

Al haberse concluido el proceso de pseudo-etiquetado y corrección, las máscaras depuradas finalmente fueron integradas al conjunto de Dataset Final. En total se obtuvieron 1005 imágenes con sus respectivas mascaras binarias en formato PNG, gracias a los procesos realizados asegurando la calidad y precisión de los datos. Por tanto, el conjunto de datos queda finalmente apto para usarse en el entrenamiento, validación y prueba de los 3 modelos de redes neuronales convolucionales a ser comparados. Por lo que, se divide de forma aleatoria los datos para tener una división del dataset de 70% para entrenamiento, 15% para validación y 15% para prueba.

Preprocesamiento de datos

Previo al entrenamiento de los 3 modelos se identificó un problema de cuello de botella durante la fase de inicialización. Al incrementarse el conjunto de datos a las 1005 imágenes con sus respectivas mascaras binarias, se experimentó un retraso significativo en el entorno de

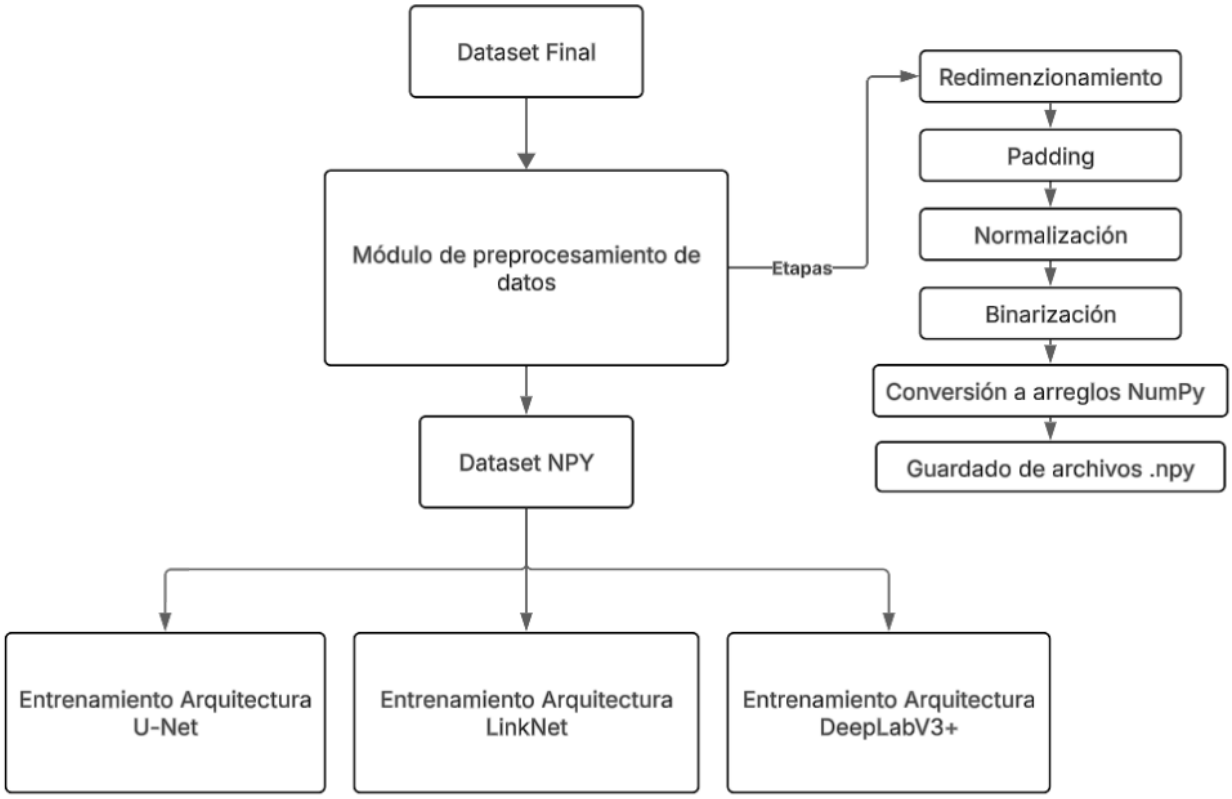
ejecución Google Colab, en donde el retardo de lectura de los datos ascendió a los 19 minutos. Mediante OpenCV se cargaba la totalidad de las imágenes en la memoria RAM previo a ejecutar cualquier operación de redimensionamiento.

Este escenario generó fallas por saturación en los canales de transferencia provocando un error en el sistema llamado *OSError*. El mensaje de error visualizado en Colab fue el siguiente: *[Errno 107] Transport endpoint is not connected,* el cual indica una desconexión temporal o perdida de sincronización con sistemas de almacenamiento remoto como lo es Google Drive. Esto suele ocurrir al haberse realizado numerosas peticiones de lecturas de archivos, tal como es el caso del presente trabajo de titulación el cual requiere de la carga del Dataset Final completo para el respectivo entrenamiento, validación y prueba de las arquitecturas.

En este sentido, procesar la carga de datos con esta demora cada vez que se iba a entrenar los modelos planificados (U-Net, LinkNet y DeepLabV3+) no era óptimo ni viable computacionalmente. Es así que se crea un cuaderno en Colab independiente para la lectura, preprocesamiento y conversión de datos hacia arreglos NumPy para así optimizar el espacio y velocidad de carga de las imágenes, con el objetivo final de consolidar una estructura de almacenamiento reproducible y estática a la cual se la llamó Dataset_NPY. Esa carpeta con los archivos .npy de las imágenes y máscaras funcionará como la única fuente de entrada optimizada para todos los experimentos de entrenamiento a realizarse. En la Figura 18 se visualiza el flujo del módulo empleado para el preprocesamiento del Dataset Final y su incidencia en el entrenamiento de las arquitecturas.

Figura 18

Flujo módulo de preprocesamiento de datos



Nota. En el diagrama se muestra que los tres modelos hacen uso del mismo dataset NPY, el cual contiene todas las imágenes y sus máscaras preprocesadas, aptas para realizar el proceso de entrenamiento, validación y prueba de U-Net, LinkNet y DeepLabV3+. Elaboración propia.

La Figura 18 presenta el flujo general del proceso de preprocesamiento implementado en el presente trabajo. A partir del conjunto de datos final se ejecutó un módulo de preprocesamiento para la estandarización de las imágenes y máscaras mediante las etapas visualizadas en la respectiva figura. Esto ayuda a optimizar el proceso de carga de los datos de entrada para los modelos que se van a entrenar y evaluar, ya que todos tendrán que usar únicamente los mismos archivos .npy, sin tener la necesidad de ejecutar tareas repetitivas para el preprocesamiento de los datos.

Redimensionamiento de imágenes y Relleno (Resize with Padding)

En el presente trabajo, como ya se había mencionado las imágenes RGB y sus respectivas máscaras tenían una resolución inicial de 4000 x 3000 píxeles. A pesar de ser una dimensión adecuada para el análisis visual, esta alta resolución aumentaría considerablemente el costo computacional al realizar el entrenamiento de las arquitecturas, por lo que, incrementaría el consumo de memoria de la GPU, la cantidad de operaciones que se deben realizar por las capas convolucionales, así como el tiempo requerido para cada época del entrenamiento.

Por tal motivo, fue necesario reducir las imágenes a una resolución de 256x256 píxeles, considerando que se está ejecutando el trabajo sobre el entorno gratuito de Google Colab, mismo que en esta versión dispone de recursos computacionales limitados. Por otro lado, se seleccionó el tamaño de 256x256 dado su amplio uso en tareas de segmentación semántica proporcionando un adecuado equilibrio entre el nivel de detalle de la información a procesar y la eficiencia computacional, además de que las CNN con estructuras de encoder-decoder trabajan mejor con imágenes cuadradas. El redimensionamiento se realizó manteniendo la proporción original de las imágenes, completando los espacios vacíos mediante la técnica de relleno de ceros conocida como *padding.* Desde la perspectiva teórica de las Redes Convolucionales, esta técnica consiste en rodear la periferia de la matriz de datos con valores nulos (ceros) con la finalidad de preservar las dimensiones espaciales de las características tras las operaciones de convolución, evitando la pérdida de información en los bordes de la imagen (Codificando Bits, 2019).

Por lo tanto, los pasos para ajustar el tamaño de las imágenes de forma automática antes de ser enviadas a los respectivos modelos de CNN son los siguientes:

1. Se encoje la imagen original de forma que el lado más largo correspondiente a los 4000 píxeles mida 256 píxeles. Por lo que, al principio, la imagen queda con un

tamaño de 256x192 debido a que las imágenes originales tenían una proporción de 4:3.

2. Para completar el espacio sobrante se aplica el *padding* de forma que para llegar a los 256 píxeles del alto se resta 256 de los 192 píxeles, dando un total de 64 píxeles que deben llenar ese espacio.
3. Finalmente se toma la imagen redimensionada de 256x192 píxeles y se la coloca justo en el centro, de forma que quedan libres 32 píxeles arriba y 32 píxeles abajo que serán rellenados con ceros mediante la función *np.zeros*. Esto se aprecia en la Figura 19.

Figura 19

Padding realizado a las imágenes y sus máscaras.



Nota. Se muestra el relleno aplicado en los píxeles superiores e inferiores de las imágenes y máscaras. Estas fueron redimensionadas a un tamaño de 256x256 píxeles, sin perder su proporción original.

Normalización

La normalización consiste en escalar los valores de intensidad de los píxeles hacia un rango común de manera que se mejore la estabilidad numérica durante el entrenamiento,

permitiendo así que se acelere el proceso de convergencia de las redes neuronales. (Akash, 2013)

En el presente trabajo, las imágenes de entrada correspondientes al canal visual (RGB) se leen con una profundidad de color de 8 bits por canal, lo que las hace tener valores discretos en el rango [0, 255]. Esto ralentiza la convergencia del algoritmo por lo que para la estandarización de las magnitudes se transformaron los arreglos al tipo de dato numérico de punto flotante de 32 bits, para luego realizar el proceso de normalización, y así obtener valores comprendidos entre 0 y 1 previo a su almacenamiento en los arreglos NumPy.

Para efectuar la normalización, cada píxel se divide entre el valor máximo de la escala, por lo que se representa esto mediante la siguiente fórmula:

$$I_{norm} = \frac{I}{255.0}$$

Binarización de máscaras

A diferencia de las imágenes de entrada, las máscaras deben trabajar de forma booleana, de manera que indiquen si hay una vía de acceso terrestre o no. Sin embargo, los procesos previos pueden llegar a introducir valores intermedios difusos en las fronteras de los polígonos que detectan dichas vías.

Para prevenir esto, se empleó un umbral a los píxeles con un valor de 127. Si el píxel tenía un valor mayor a 127, entonces se lo asignará con un valor 1 indicando la presencia de la clase carretera sin inundación, mientras que si el valor del pixel es igual o menor a 127 se asignará 0 indicando que no hay vías. Así se elimina cualquier ruido o degradado gris no deseado. En la siguiente fórmula se expresa rápidamente lo mencionado.

$$M(x,y) = \begin{cases} 1, & I(x,y) > 127 \\ 0, & I(x,y) \leq 127 \end{cases}$$

Generación de archivos NumPy

Una vez completadas las etapas anteriores, las imágenes y máscaras fueron convertidas en arreglos multidimensionales de NumPy y se guardaron en archivos con

extensión .npy, respetando la división previa de los subconjuntos de entrenamiento (70%), validación (15%) y prueba (15%). Este nuevo formato en que se guardaron los datos permitió que se carguen de forma directa en memoria con la instrucción np.load(), omitiendo el flujo repetitivo de lectura y conversión en cada corrida por lo que la transferencia de datos se volvió prácticamente instantánea: El tiempo de carga del conjunto de datos completo pasó de 19 minutos a tan solo unos segundos. Esta estandarización en NumPy aseguró que U-Net, LinkNet y DeepLabV3+ procesaran exactamente el mismo conjunto de datos sin discrepancias.

Configuración del entrenamiento

Tras finalizar la construcción del conjunto de datos y su respectivo preprocesamiento, se procedió a establecer los detalles necesarios para asegurar un entrenamiento consistente y reproducible para las tres arquitecturas que se van a probar en el presente trabajo. Esto incluyó preparar los datos, definir los parámetros generales para el entrenamiento, elegir las funciones adecuadas para la optimización y los métodos de seguimiento para supervisar el aprendizaje y guardar el mejor rendimiento de los modelos.

Preparación de los datos

Previo al entrenamiento de cada una de las arquitecturas, se estandarizó la preparación de los datos mediante las siguientes funciones:

Data Augmentation

A fin de enriquecer la variabilidad del conjunto de datos y prevenir el sobreajuste del modelo, se recurrió a diversas técnicas de aumentación de datos. El *Data Augmentation* consiste en generar nuevas muestras de entrenamiento mediante transformaciones aplicadas sobre las imágenes originales, con el propósito de incrementar la variabilidad del conjunto de datos y mejorar la capacidad de generalización del modelo (Shorten & Khoshgoftaar, 2019).

En el presente trabajo, el aumento de datos se aplicó únicamente al conjunto de entrenamiento para aumentar la variabilidad de las muestras disponibles y mejorar la capacidad de generalización de los modelos. Para ello se utilizaron transformaciones aleatorias como

rotaciones de 90°, inversiones horizontal y vertical, además de pequeñas variaciones en el brillo de las imágenes. Esto se aplicó tanto a las imágenes como a sus respectivas máscaras mediante *TensorFlow* y ejecutadas de forma probabilística durante el entrenamiento.

Dataset TensorFlow

Una vez generado el conjunto de datos preprocesado, los arreglos almacenados en formato *NumPy* (.npy) fueron cargados mediante la API *tf.data.Dataset* de *TensorFlow* para optimizar el flujo de lectura durante el entrenamiento. Posteriormente, los datos fueron organizados en lotes (*batches*) y preparados para su procesamiento por cada una de las tres arquitecturas implementadas.

Parámetros generales

En la tabla 2 se visualiza de forma general los parámetros establecidos para el entrenamiento de las arquitecturas.

Tabla 2

Valores iniciales establecidos para los parámetros de entrenamiento de las arquitecturas

Parámetro	**Valor establecido**
Tamaño de imagen	256 × 256
Batch Size	8
Learning Rate inicial	0.001
Optimizador	Adam
Máximo de épocas	70

Nota. Elaboración propia.

Batch Size

El Batch Size corresponde al número de imágenes procesadas en la red neuronal antes de realizar una actualización de los pesos del modelo, es decir, sus parámetros internos. En el presente trabajo se utilizó un tamaño de lote de ocho imágenes, debido a que permitió aprovechar adecuadamente la memoria disponible de la GPU NVIDIA Tesla T4 empleada durante el entrenamiento y así no sobrecargar la memoria. Cabe destacar que un tamaño de lote de ocho fue suficiente para optimizar la actualización de los pesos del modelo, cuidando así que se genere un sobreajuste de los modelos a entrenar.

Optimizer (Adam)

Adam es un algoritmo de optimización que se utiliza mucho en el aprendizaje profundo. Este algoritmo ajusta automáticamente la velocidad a la que el modelo aprende durante el entrenamiento (Kingma & Lei Ba, 2017). En esta investigación, inicialmente se empleó el optimizador Adam con los parámetros predeterminados de TensorFlow para actualizar los pesos de las redes neuronales.

Learning Rate inicial

La tasa de aprendizaje inicial define el tamaño de las actualizaciones realizadas sobre los pesos de la red durante el entrenamiento. En este trabajo se estableció un valor inicial de 0.001, permitiendo una convergencia estable durante las primeras épocas de entrenamiento. Sin embargo, luego se optó por emplear un learning rate de 0.0001, de forma que se reduce 10 veces el tamaño del paso que da el modelo de entrenamiento. Esto se realizó con el fin de obtener afinación más precisa y mayor estabilidad, a pesar de requerirse más épocas para converger.

Número máximo de épocas

El entrenamiento se configuró con un máximo de 70 épocas. Sin embargo, la cantidad real de épocas que se ejecutaron al entrenar cada una de las arquitecturas dependió de cómo se comportaba la función de pérdida implementada, ya que el proceso fue supervisado

mediante el *callback EarlyStopping*, que evita la realización de entrenamientos innecesarios cuando los modelos ya no mejoraban.

Funciones de pérdida empleadas en el entrenamiento

Durante el entrenamiento de modelos de segmentación semántica se requiere definir funciones que orienten el proceso de aprendizaje y permitan medir la diferencia entre las predicciones generadas por la red neuronal y las máscaras de referencia. Estas funciones proporcionan la información necesaria para el ajuste de los parámetros del modelo en las respectivas iteraciones con el fin de obtener la mejor respuesta en segmentación. (Jadon, 2020).

Dice Coefficient (como métrica)

El *Dice Coefficient* es una métrica muy usada en tareas de segmentación semántica, buscando medir la concordancia que existe entre la máscara estimada por el modelo y la de referencia. Arroja valores entre 0 y 1, donde valores cercanos a 1 indican una mayor superposición entre ambas segmentaciones (Jadon, 2020). En este trabajo, el *Dice Coefficient* se empleó como métrica de seguimiento durante el entrenamiento para monitorear el desempeño de las arquitecturas implementadas.

Dice Loss (función de pérdida inicial)

La *Dice Loss* corresponde a una adaptación del *Dice Coefficient* utilizada como función de pérdida para problemas de segmentación semántica. Su objetivo es reducir la diferencia entre la segmentación predicha y la máscara real, y de esa manera ayuda a que se genere una mayor superposición entre ambas durante el proceso de aprendizaje (Jadon, 2020). En las primeras etapas del desarrollo de esta investigación, esta función fue utilizada para optimizar el entrenamiento de los modelos implementados.

Binary Cross-Entropy + Dice Loss

La *Binary Cross-Entropy* (BCE) mide la diferencia entre la probabilidad predicha por el modelo y la clase real de cada píxel.(Jadon, 2020). Para este trabajo se empleó una función

híbrida que combina *Binary Cross-Entropy* y *Dice Loss*, permitiendo considerar tanto la clasificación individual de cada píxel como la superposición global entre las máscaras predichas y las reales, obteniendo un entrenamiento más estable y un mejor desempeño de los modelos. Es por esto que usar ambas a la vez permite que mediante BCE se clasifique correctamente cada píxel y mediante la función *Dice* se mantiene la forma general de la estructura.

Callbacks

Los *Callbacks* son funciones que se ejecutan automáticamente durante diferentes etapas del entrenamiento de una red neuronal. Estas funciones permiten monitorear el comportamiento del modelo y actuar en base a requerimientos necesarios para la mejor convergencia y desempeño del mismo. A continuación, se explican las funciones empleadas para las tres arquitecturas.

EarlyStopping

EarlyStopping monitorea el desempeño del modelo durante el entrenamiento y detiene automáticamente el proceso cuando la métrica evaluada deja de mejorar después de cierto número determinado de épocas (GeeksforGeeks, 2025). Esto fue muy útil para evitar el riesgo de sobreajuste, ya que se apagaba el entrenamiento por completo si el error *val_loss* dejaba de mejorar durante 10 épocas seguidas.

ReduceLROnPlateau

ReduceLROnPlateau disminuye automáticamente la tasa de aprendizaje cuando el desempeño del modelo permanece sin mejoras durante varias épocas consecutivas (GeeksforGeeks, 2025). Esto ayuda a obtener una convergencia más estable de los modelos que se están entrenando, dado que reducía el *learning rate* a la mitad gracias al factor de 0.5, dando así pasos más finos y pequeños para encontrar el mínimo error posible.

ModelCheckpoint

ModelCheckpoint permite almacenar automáticamente los pesos del modelo que obtienen el mejor desempeño sobre el conjunto de validación (GeeksforGeeks, 2025). En esta investigación se empleó para conservar los datos de la mejor época de cada arquitectura entrenada y guardar así los modelos en Google Drive, permitiendo guardar las mejores versiones.

Implementación de modelos

Con el propósito de identificar las zonas de acceso terrestre en escenarios posteriores a una inundación, en el presente trabajo de titulación se implementaron tres arquitecturas de segmentación semántica ampliamente utilizadas en tareas de visión por computador: U-Net, LinkNet y DeepLabV3+. Estos modelos fueron seleccionados debido a su excelencia para realizar segmentación a nivel de píxel, así como para tareas enfocadas a la detección de objetos en imágenes. Para cada arquitectura se mantuvo una configuración de entrenamiento consistente, de forma que se realizó una comparación objetiva de su desempeño en la identificación de vías terrestres transitables dentro del conjunto de datos empleado. En los siguientes apartados se describe la implementación de cada modelo, así como los principales ajustes realizados durante el proceso de entrenamiento y optimización, hasta llegar a la obtención de los modelos finales.

Implementación de U-Net

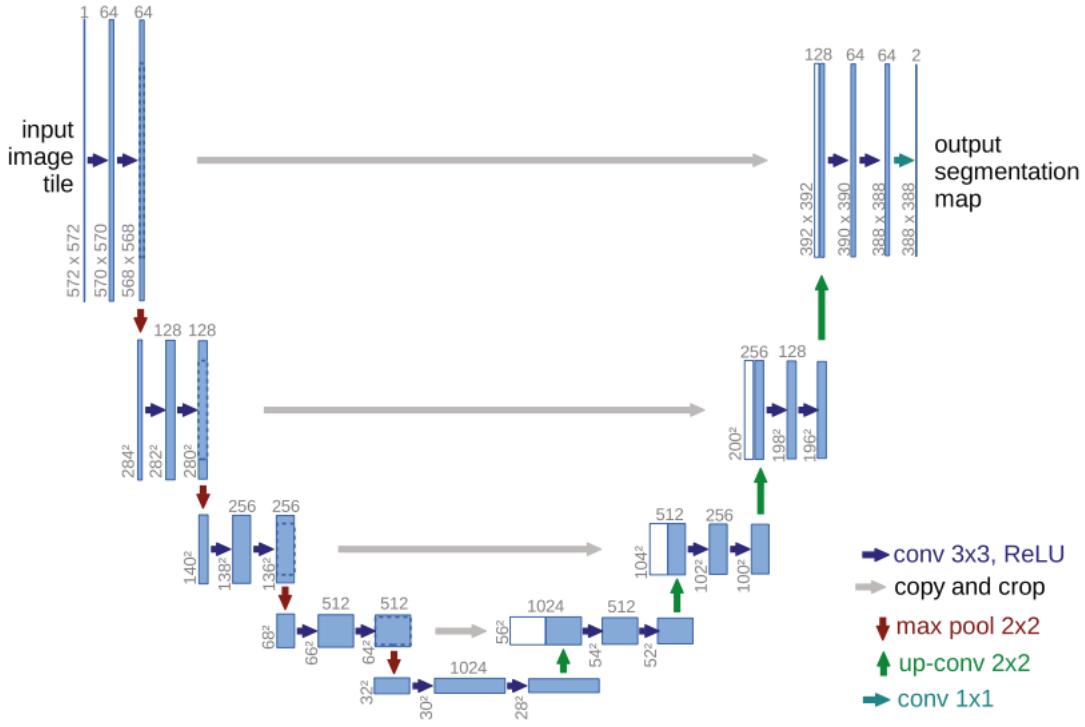
La arquitectura U-Net es una red neuronal convolucional diseñada originalmente para la segmentación de imágenes biomédicas, sin embargo, mostró una alta eficacia para preservar la resolución espacial en distintas escalas por lo que también se trasladó su uso a la teledetección. La arquitectura destaca por una topología simétrica en "U", donde el bloque de codificación se equilibra directamente con la fase de expansión (Ronneberger et al., 2015).

Arquitectura implementada

La arquitectura U-Net, propuesta por Ronneberger et al. (2015), fue diseñada para tareas de segmentación semántica, de manera que se pueda realizar una clasificación a nivel de píxel mediante una estructura tipo *encoder-decoder* con conexiones de salto. Su diseño facilita la extracción de características importantes en la etapa de codificación y la recuperación progresiva de la resolución espacial durante la decodificación. Esto ayuda a mantener la información relevante proveniente de las primeras capas de la red. La arquitectura general empleada se presenta en la Figura 20.

Figura 20

Arquitectura U-Net



Nota. Los bloques azules representan mapas de características multicanal, donde el número de canales se indica en la parte superior y el tamaño espacial (x-y) aparece en la esquina inferior izquierda. Los bloques blancos corresponden a los mapas de características copiados, mientras

que las flechas señalan las distintas operaciones realizadas. Tomado de (Ronneberger et al., 2015).

Originalmente, el diseño de la ruta de contracción está formada por bloques repetitivos compuestos por dos capas convolucionales de 3×3, cada una seguida por una función de activación ReLU, mientras que la reducción de la resolución espacial se realiza mediante operaciones *Max Pooling* de 2×2 con *stride* igual a 2. A medida que disminuyen las dimensiones espaciales de los mapas de características, el número de filtros se incrementa progresivamente, es así que se capturan representaciones cada vez más complejas de la imagen. Después de eso, la ruta expansiva va mejorando la resolución poco a poco. Para hacerlo, utiliza operaciones de *upsampling* y combina la información que obtuvo del codificador mediante conexiones de salto. Luego, aplica nuevamente bloques convolucionales. La ruta expansiva combina la información del codificador con las conexiones de salto antes de aplicar los bloques convolucionales. Finalmente, una convolución 1×1 genera el mapa de segmentación correspondiente a la salida del modelo (Ronneberger et al., 2015).

Para el presente trabajo de titulación, la arquitectura fue implementada en TensorFlow/Keras realizando algunas adaptaciones respecto al modelo original. Se emplearon bloques convolucionales constituidos por dos capas *Conv2D* para la extracción de características importantes, en conjunto con la función de activación ReLU, además de usar *padding 'same'* para conservar las dimensiones espaciales de los mapas de características. Para la estructura del codificador se usaron 32, 64, 128 y 256 filtros, mientras que el cuello de botella utilizó 512 filtros junto con una capa Dropout para reducir la probabilidad de tener sobreajuste. En el decodificador se utilizaron operaciones *UpSampling2D* para así aumentar las dimensiones espaciales, seguidas de la concatenación con los mapas provenientes del codificador y nuevos bloques convolucionales, finalizando con una capa *Conv2D* de 1×1 con activación *Sigmoid* que permite generar la máscara binaria correspondiente a las vías terrestres transitables ante los escenarios de post inundación.

Optimización de hiperparámetros

Una vez implementada la arquitectura base de U-Net, se realizaron diferentes experimentos a fin de mejorar su capacidad de segmentación. Para ello se modificaron algunos hiperparámetros relacionados con el proceso de entrenamiento, evaluando el desempeño de cada configuración mediante el coeficiente Dice obtenido sobre el conjunto de validación. Los hiperparámetros analizados fueron la función de pérdida, el valor de dropout, el número máximo de épocas y el parámetro patience del criterio de detención temprana que es uno de las operaciones de *callback*. La Tabla 3 resume las configuraciones evaluadas durante esta etapa.

Tabla 3

Experimentos realizados para la optimización de U-Net.

Versión	Dice Loss	BCE+Dice	Dropout	Épocas	Patience	Dice Validación
V1	✓	—	0.3	50	10	0.726
V2	—	✓	0.3	50	10	0.815
V3	—	✓	0.4	50	10	**0.848**
V4	—	✓	0.4	70	15	0.826

Nota. Elaboración propia.

De la Tabla 3 se puede visualizar que en la primera versión se utilizó únicamente Dice Loss como función de pérdida; sin embargo, el desempeño obtenido no fue satisfactorio, por lo que esta configuración fue descartada. Posteriormente se incorporó una función de pérdida combinada BCE + Dice Loss, observándose una mejora importante en el coeficiente Dice sobre el conjunto de validación.

En una tercera versión se incrementó el valor de dropout de 0.3 a 0.4 con el fin de potenciar la solidez modelo y reducir el riesgo de sobreajuste. Esta configuración obtuvo el mejor desempeño durante la etapa de validación, alcanzando un Dice de 0.848.

Finalmente, se realizó un último experimento aumentando el número máximo de épocas de entrenamiento a 70 y la paciencia del criterio de detención temprana a 15, con la idea de que a más épocas de entrenamiento mejores resultados se obtendrían ante el entrenamiento del modelo. No obstante, esta modificación produjo una disminución en el coeficiente Dice de validación, por lo que se decidió conservar la configuración de la versión 3 como modelo definitivo.

Configuración final del modelo

De acuerdo con los resultados obtenidos durante la etapa de optimización, se seleccionó la configuración correspondiente al tercer experimento para el entrenamiento definitivo de la arquitectura U-Net. Esta configuración tuvo el mejor rendimiento en el conjunto de validación. Obtuvo el coeficiente Dice más alto y un comportamiento más estable durante el entrenamiento. De igual forma, durante la etapa de codificación, U-Net emplea operaciones de *Max Pooling* para reducir progresivamente la resolución de los mapas de características, permitiendo extraer información de mayor nivel semántico.

En la Tabla 4 se resumen los principales parámetros que conforman la configuración final implementada en el presente trabajo.

Tabla 4

Configuración final del modelo U-Net

Parámetro	**Valor**
Filtros encoder	32-64-128-256
Bridge	512 filtros

Kernel	3×3
Padding	Same
Activación	ReLU
Batch Normalization	Sí
MaxPooling	2×2
UpSampling	2×2
Skip Connections	Concatenación
Dropout	0.4
Salida	Sigmoid
Función pérdida	BCE + Dice Loss

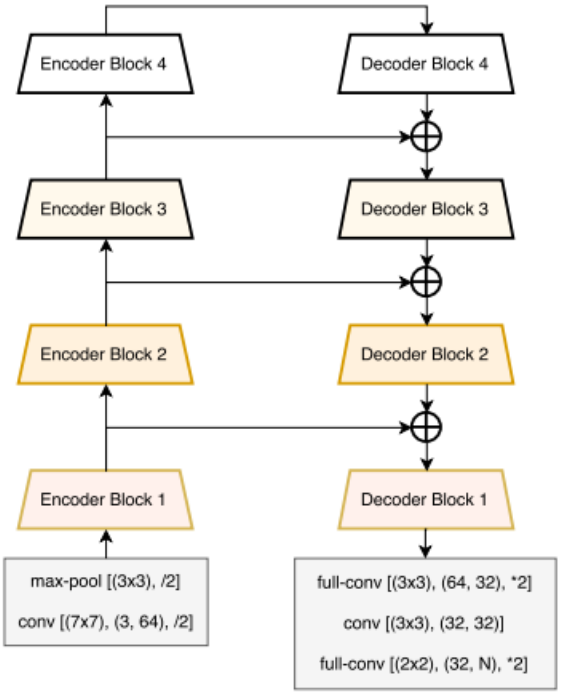
Nota. Elaboración propia.

Implementación de LinkNet

LinkNet es una red de segmentación semántica ligera y eficiente en términos de procesamiento. A diferencia de otras arquitecturas, destaca por conectar de manera directa y rápida las capas del codificador (*encoder*) con el decodificador (*decoder*) sumando sus mapas de características. Por tal razón, a diferencia de U-Net, las conexiones entre ambas rutas se realizan mediante una operación de suma (*summation*), lo que favorece la reutilización de características y reduce la complejidad computacional del modelo, así como el tiempo de procesamiento sin perder precisión en los contornos de los objetos (Chaurasia & Culurciello, 2017). En la Figura 21 se observa la arquitectura general de LinkNet.

Figura 21

Arquitectura LinkNet



Nota. Tomado de (Chaurasia & Culurciello, 2017).

Arquitectura implementada

Para el presente trabajo, la arquitectura LinkNet fue implementada utilizando TensorFlow/Keras, conservando la estructura general propuesta en el modelo original y adaptándola al problema de segmentación binaria de vías terrestres. La red recibe imágenes RGB de 256 × 256 píxeles como entrada y genera una máscara binaria mediante una capa convolucional final de 1 × 1 con función de activación *Sigmoid*.

En el diseño del codificador se integraron bloques residuales cuyos canales aumentan progresivamente (64, 128, 256, 512 y 1024). Las operaciones convolucionales emplearon un *kernel* de 3x3 junto con *Batch Normalization* y la función ReLU. La reducción dimensional de la imagen se logró aplicando un *stride* de 2 en las convoluciones, lo que favorece el aprendizaje de representaciones complejas a medida que va haciéndose más profunda la red.

El decodificador se implementó utilizando capas *Conv2DTranspose* para realizar el sobremuestreo de los mapas de características. Después de cada operación de reconstrucción, las características provenientes del codificador fueron integradas mediante una operación de suma, característica principal de LinkNet, seguida por una convolución de 3 × 3, B*atch Normalization* y activación ReLU para refinar la información antes de la siguiente etapa de reconstrucción.

Finalmente, la salida del modelo se obtuvo mediante una convolución de 1 × 1 con activación Sigmoid, permitiendo estimar para cada píxel la probabilidad de pertenecer a la clase correspondiente a las vías terrestres no inundadas.

Optimización de hiperparámetros

Con el propósito de mejorar el desempeño del modelo, se realizaron distintos experimentos modificando algunos hiperparámetros del entrenamiento. En cada experimento se evaluó la evolución de las curvas de aprendizaje y las métricas obtenidas sobre el conjunto de validación, de forma que finalmente se realizó la selección de la configuración que presentó el mejor equilibrio entre capacidad de aprendizaje y estabilidad durante el entrenamiento.

Tabla 5

Experimentos realizados para la optimización de LinkNet.

Versión	Learning Rate	Épocas
V1	0.0001	50
V2	0.0005	70

Nota. Elaboración propia.

Como se aprecia en la Tabla 5, inicialmente el modelo fue entrenado utilizando una tasa de aprendizaje de 0.0001. Durante este experimento se observó una convergencia estable en las etapas finales del entrenamiento, sin evidencias claras de sobreajuste, ya que las curvas

correspondientes al conjunto de entrenamiento y validación presentaban un comportamiento similar. Sin embargo, durante las épocas iniciales se registraron variaciones notables en la métrica de validación lo que indicó que se tenía un aprendizaje inestable al comenzar el entrenamiento.

Inicialmente se fijó el *learning rate* en 0.0001, lo cual garantizó estabilidad en las primeras etapas. Pero ante la evidencia gráfica de que el modelo mantenía margen de mejora se decidió elevar el valor a 0.0005 y el número máximo de épocas a 70. Aunque durante el entrenamiento se observó un incremento puntual en la pérdida de validación, las métricas finales mejoraron significativamente respecto a la configuración inicial, motivo por el cual esta versión fue seleccionada como modelo definitivo.

Gracias a la última configuración, se obtuvo una mejora en las métricas de validación durante el entrenamiento, especialmente en el coeficiente Dice. Aunque a lo largo del entrenamiento se presentó un incremento puntual en la pérdida de validación, el comportamiento general de las curvas y los resultados obtenidos evidenciaron un mejor desempeño respecto a la configuración inicial, motivo por el cual esta versión donde se modifica el *learning rate* fue seleccionada como modelo definitivo.

Configuración final del modelo

Después del proceso de experimentación previamente mencionado, se definió una configuración final para la arquitectura LinkNet, correspondiente a la versión que obtuvo el mejor desempeño sobre el conjunto de validación. Los principales parámetros utilizados durante el entrenamiento se resumen en la Tabla 6.

Tabla 6

Configuración final del modelo LinkNet

Parámetro	Valor
Bloques residuales	64, 128, 256, 512 y 1024 filtros
Decoder	Conv2DTranspose + Add
Kernel	3 × 3
Función de activación	ReLU
Salida	Conv2D 1 × 1 + Sigmoid
Learning Rate	0.0005
Batch Size	8
Función de pérdida	BCE + Dice Loss
Optimizador	Adam
Épocas máximas	70

Nota. Elaboración propia.

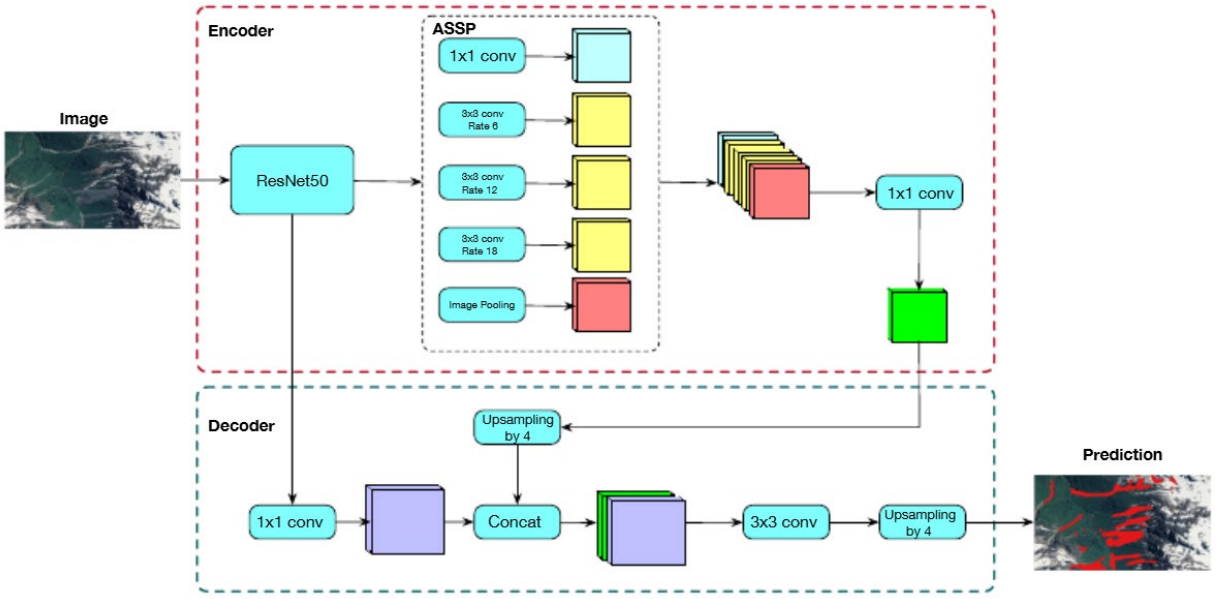
Implementación de DeepLabV3+

DeepLabV3+ es una arquitectura de segmentación semántica propuesta por Chen et al. (2018). Esta fue diseñada para mejorar la precisión en la delimitación de los objetos segmentados mediante la combinación de un codificador encargado de extraer información contextual y un decodificador responsable de recuperar los detalles espaciales perdidos durante el proceso de extracción de características, además de capturar el contexto de los objetos con mucha precisión a diferentes escalas. A diferencia de DeepLabV3, esta versión

incorpora un decodificador que refina los bordes de la segmentación utilizando información de bajo nivel proveniente del *backbone*. Es así que este modelo obtiene máscaras más precisas especialmente en los límites de los objetos (Si et al., 2021). En la Figura 22 se visualiza la arquitectura general de DeepLabV3+.

Figura 22

Arquitectura DeepLabV3+



Nota. Tomado de (Morales et al., 2022)

En base a la Figura 22 se describe a continuación algunos de los componentes básicos para la constitución de la presente arquitectura.

- **Encoder–Decoder**

El codificador extrae características semánticas de alto nivel a partir de la imagen de entrada. Para ello emplea un *backbone* convolucional encargado de reducir progresivamente la resolución espacial a medida que se incrementa la cantidad de información representada en los mapas de características. Luego, estas representaciones son procesadas mediante el módulo ASPP que va a permitir incorporar información contextual proveniente de diferentes escalas (Chen et al., 2018).

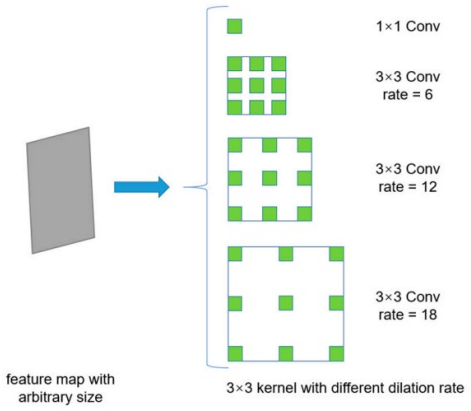
El módulo decodificador se encarga de reconstruir gradualmente la resolución espacial de la imagen segmentada mediante operaciones de sobremuestreo e integración de características de bajo nivel obtenidas en las primeras capas del *backbone*. De esa forma, la combinación de *encoder* y *decoder* reconstruye con mayor precisión los contornos de los objetos segmentados, compensando la pérdida de detalles ocasionada durante la etapa de codificación (Chen et al., 2018).

- **Módulo ASPP**

Uno de los componentes principales de DeepLabV3+ corresponde al módulo *Atrous Spatial Pyramid Pooling*. El objetivo del presente módulo es capturar información contextual de la imagen considerando diferentes escalas simultáneamente. Para ello aplica varias convoluciones *atrous* con distintas tasas de dilatación sobre un mismo mapa de características, además de una rama de *Average Pooling* global. Finalmente, todas las salidas son concatenadas para la generación de una representación multiescala más completa. Esto va a favorecer la segmentación de objetos con diferentes tamaños y formas (Si et al., 2021). En la Figura 23 se observa la estructura del módulo ASPP que se emplea en la arquitectura de DeepLabV3+.

Figura 23

Estructura ASPP en DeepLabV3+



Nota. Tomado de (Si et al., 2021).

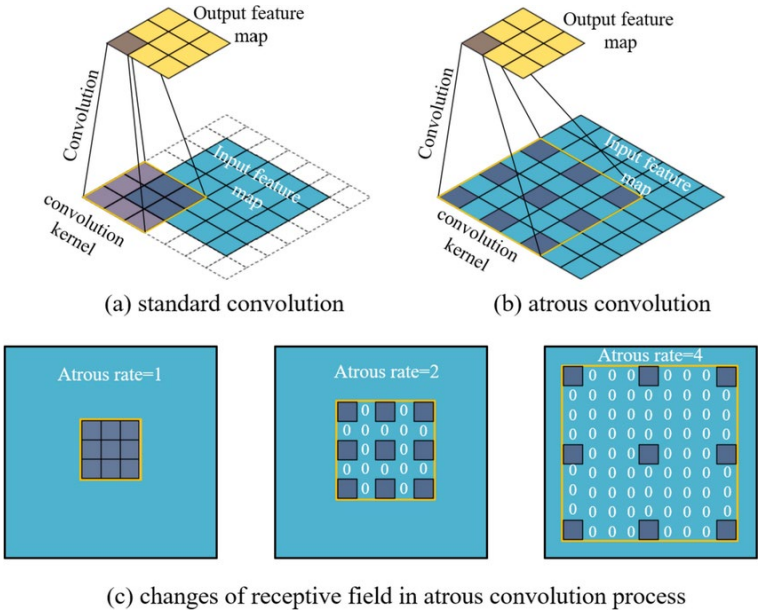
Por tanto, a diferencia de las arquitecturas anteriores, DeepLabV3+ incorpora la operación *Average Pooling* dentro del módulo ASPP. Gracias a esta operación de obtener los valores promedio, se suaviza la representación, de forma que permite capturar información del contexto global de la imagen antes del proceso de decodificación.

- **Convolución Atrous**

La convolución atrous, conocida también como convolución dilatada, consiste en insertar espacios entre los elementos del filtro convolucional con el fin de ampliar el campo receptivo sin aumentar el número de parámetros ni disminuir la resolución del mapa de características. Gracias a esto, se captura información sobre una parte más grande de la imagen sin perder detalle. Esto es importante especialmente en tareas de segmentación semántica donde la localización precisa de los objetos es fundamental (Chen et al., 2018). Se aprecia la diferencia de una convolución estándar a la dilatada en la Figura 24.

Figura 24

Convolución dilatada



Nota. a) Convolución estándar con *Kernel* 3x3 y *padding*, b) Convolución atrous con kernel 3x3 y atrous rate de 2, c) convolución atrous se realiza de forma independiente a tasas de expansión

de 1, 2 y 4; convolución kernel no ocupa una región rellena de ceros. Tomado de (G. Li et al., 2024).

En DeepLabV3+, la convolución *atrous* no actúa como una función aislada, sino que transforma la estructura interna de la red ResNet50, la cual se desempeña como red base o *backbone* para la extracción de características (Tomar, 2021).

Arquitectura implementada

Para el presente trabajo de titulación se implementó la arquitectura DeepLabV3+ utilizando TensorFlow/Keras, empleando como codificador un backbone *ResNet50* preentrenado con pesos de *ImageNet*, encargado de la extracción de representaciones jerárquicas de la imagen de entrada. A partir de las características profundas generadas por el *backbone* se incorporó el módulo ASPP. Este módulo se conformó de una rama de *Average Pooling* global y cuatro convoluciones con tasas de dilatación 1, 6, 12 y 18, permitiendo capturar información contextual a diferentes escalas y aumentar el campo receptivo sin perder resolución espacial en los mapas de características.

Gracias al módulo ASPP, el modelo mejora la comprensión de estructuras de gran tamaño y favorece una segmentación más consistente en escenarios complejos tal como la búsqueda de zonas o caminos de acceso terrestre en sitios post inundación.

Posteriormente, las características obtenidas por el ASPP fueron ampliadas mediante interpolación bilineal e integradas con características de bajo nivel provenientes de las primeras capas del *backbone* mediante una operación de concatenación. Esta estrategia combina información semántica detallada con datos espaciales de alta resolución. Así se consiguen delimitar mejor los bordes de las carreteras. Finalmente, se aplicaron dos capas convolucionales de 256 filtros, acompañadas por *Batch Normalization* y activación ReLU, obteniendo como salida una máscara binaria mediante una convolución 1×1 con función de activación *Sigmoid*.

Configuración final del modelo

Una vez definida la arquitectura, se estableció la configuración final del modelo DeepLabV3+, cuyos principales componentes y parámetros utilizados durante el entrenamiento se resumen en la Tabla 7.

Tabla 7

Configuración final del modelo DeepLabV3+

Parámetro	**Valor**
Tamaño de entrada	256 × 256 × 3
Backbone	ResNet50 (ImageNet)
Encoder	ResNet50
ASPP	Average Pooling + dilataciones 1, 6, 12 y 18
Decoder	Upsampling bilineal + concatenación
Activación intermedia	ReLU
Batch Normalization	Sí
Salida	Conv2D 1×1 + Sigmoid

Nota. Elaboración propia.

Capítulo IV: Resultados y análisis

En este capítulo se presentan y discuten los resultados obtenidos tras la fase de experimentación y entrenamiento. El propósito es evaluar el rendimiento de los modelos desarrollados e identificar el que posea mejores resultados para la detección de vías de acceso terrestre en escenarios post inundación. Con base en estos resultados, se implementa el módulo de segmentación utilizando el modelo que obtuvo el mejor desempeño.

Métricas de evaluación del rendimiento

Las métricas de evaluación son fundamentales para el análisis del desempeño de modelos de segmentación semántica, pues de manera objetiva, permite la cuantificación de la capacidad del algoritmo en la identificación y delimitación correcta de las regiones de interés dentro de una imagen. De esta manera, la evaluación se basa en la comparación entre el mapa de segmentación real o Ground Truth (G), que representa la referencia verdadera de las clases presentes, y el mapa de segmentación predicho (P), el cual es generado por los modelos durante la inferencia. Por ende, a partir de la relación entre predicción y referencia, es posible el cálculo de indicadores que miden la precisión espacial, sensibilidad y confiabilidad del sistema, puntos relevantes en aplicaciones críticas como la detección de zonas de acceso terrestre en entornos de desastres naturales (Cando Garces, 2018).

- **Matriz de confusión**

La matriz de confusión se utiliza para la evaluación del rendimiento de un modelo de clasificación o segmentación mediante la comparación entre las predicciones generadas y las etiquetas reales. Referente a la segmentación de imágenes, cada píxel es tratado como una unidad de clasificación, por ende, la matriz resume el número de pixeles clasificados de forma correcta e incorrecta para cada clase. En este sentido, esta representación facilita la visualización del comportamiento del modelo durante la etapa de evaluación y constituye la base para el cálculo de métricas como *accuracy*, *precision*, *recall*, F1 y especificidad (Borja et al., 2020).

De tal manera, la matriz de confusión se conforma por los verdaderos positivos (VP), que representan los pixeles positivos identificados correctamente por el modelo, además, los verdaderos negativos (VN), correspondientes a los pixeles negativos clasificados correctamente, también, los falsos positivos (FP) que ocurren cuando un pixel negativo es clasificado como positivo, finalmente, los falsos negativos (FN) que se presentan cuando un pixel positivo es clasificado como negativo. Se reconoce que los FP corresponden a errores de tipo I y los FN a errores de tipo II, y su magnitud permite identificar si el modelo tiende a la generación de detecciones incorrectas o, por el contrario, a omitir regiones que realmente pertenecen a la clase de interés (García Vega, 2024).

A continuación, se muestra la representación de la matriz de confusión en la Figura 25.

Figura 25

Matriz de confusión



Nota. Tomado de (AprendeIA, 2019).

- **Dice**

El coeficiente de similitud de DICE (DSC) es una métrica de validación estadística muy común en la evaluación del rendimiento de modelos de segmentación de imágenes; gracias a su capacidad para medir tanto la reproducibilidad de segmentaciones manuales, al igual que la precisión de la superposición espacial entre regiones segmentadas de forma automática. Este coeficiente permite cuantificar el nivel de similitud entre el mapa de segmentación real (G) y el

mapa predicho (P), y expresa el grado de coincidencia como un porcentaje de superposición entre ambas máscaras (Zou et al., 2004). En el contexto de redes neuronales convolucionales aplicadas a segmentación semántica, esta métrica resulta esencial porque permite valorar con mayor sensibilidad la calidad de la delimitación de zonas específicas dentro de una imagen.

Su fórmula se expresa como:

$$DICE = \frac{2(G \cap P)}{|G| + |P|}$$

En donde la intersección $G \cap P$ corresponde a los casos de verdaderos positivos, el total de los reales $|G|$ corresponde a la suma de verdaderos positivos más falsos negativos y el total predicho $|P|$ concierne a la suma entre verdaderos positivos y los falsos positivos. Por tanto, la fórmula también puede expresarse de la siguiente manera:

$$DICE = \frac{2(VP)}{2(VP) + FP + FN}$$

- **IoU**

La métrica de *Intersection over Union* (IoU) en la segmentación semántica sirve para la cuantificación del porcentaje o nivel de concordancia entre la máscara real y la salida generada por el modelo, permitiendo evaluar de manera precisa la correspondencia espacial entre ambas regiones. Por su parte, la fórmula de IoU representa la relación entre la intersección y la unión de los pixeles pertenecientes al mapa de referencia (G) y al mapa predicho (P), a través del cálculo del número de pixeles comunes dividido por el número total de pixeles presentes en ambas máscaras (Astudillo & Revelo, 2021)

A continuación, se muestra la fórmula utilizada para la métrica IoU:

$$IoU = \frac{Intersección}{Unión} = \frac{VP}{VP + FP + FN}$$

- **Accuracy**

La métrica de *accuracy* o exactitud expresa la proporción de predicciones correctas en relación al total de predicciones realizadas por el modelo, de manera que, establece una

relación directa entre los aciertos obtenidos y las etiquetas reales presentes en el conjunto de evaluación. En este sentido, un valor de exactitud de 1 indica que existe una coincidencia total entre las predicciones generadas y el mapa de segmentación real, lo que indica un desempeño ideal en el proceso de clasificación o segmentación. Sin embargo, está métrica presenta una disminución en su utilidad en escenarios donde existe un gran desequilibrio de clases, como en casos en los que una categoría predomina ampliamente sobre las demás, por ende, su análisis suele complementarse con métricas que describen con mayor detalle el comportamiento del modelo (Vivanco Toala & Camero Berrones, 2025).

La fórmula de la métrica de accuracy es:

$$Accuracy = \frac{VP + VN}{VP + FP + FN + VN}$$

- **Precision**

La métrica *precision* define la proporción de predicciones verdaderamente positivas con respecto al total de casos que el modelo ha clasificado como positivos, reconociendo que tan acertadas son las detecciones realizadas durante el proceso de evaluación. Su valor aumenta a medida que disminuyen los falsos positivos, lo que hace referencia a que el modelo comete menos errores al identificar elementos pertenecientes a la clase de interés (Vivanco Toala & Camero Berrones, 2025). Ahora bien, en el caso de la segmentación de imágenes aéreas, la métrica de *precision* ofrece información sobre el nivel de confiabilidad de las regiones detectadas como zonas de acceso terrestre frente a escenarios donde pueden existir elementos visuales similares.

La fórmula de esta métrica es:

$$Precision = \frac{VP}{VP + FP}$$

- **Recall**

La métrica de *recall* expresa la capacidad de un modelo para identificar de forma correcta los casos positivos reales dentro de un conjunto de datos, a partir de la relación de los

verdaderos positivos detectados con el total de instancias que realmente pertenecen a esa clase. Así pues, un valor cercano a 1 indica que el modelo logra el reconocimiento de la mayoría de elementos positivos, y alcanza un nivel alto de cobertura sobre la información de interés presente en la imagen (Vivanco Toala & Camero Berrones, 2025). Además, su uso cobra mayor importancia en contextos donde el objetivo primario es mitigar la presencia de falsos negativos, como es el caso de la detección de zonas de acceso terrestre en escenarios de desastre, donde la omisión de una ruta viable podría afectar la toma de decisiones operativas.

Se expresa como:

$$Recall = \frac{VP}{VP + FN}$$

- **F1**

La métrica F1 corresponde al promedio ponderado entre precision y recall, proporcionando una medida equilibrada del desempeño del modelo cuando ambas métricas tienen una misma relevancia durante la evaluación. De este modo, al considerar tanto los falsos positivos como los falsos negativos, esta métrica otorga una visión más representativa del comportamiento del modelo que la obtenida al analizar cada métrica por separado. Por esta razón, su aplicación es especialmente útil en problemas con desbalances de clases, como la segmentación de imágenes, porque permite evaluar de manera conjunta la capacidad del modelo para detectar correctamente las regiones de interés y, de igual manera, reducir las predicciones erróneas (Vivanco Toala & Camero Berrones, 2025).

La fórmula de la métrica F1 es:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

- **Especificidad**

La especificidad también denominada tasa verdadera negativa, mide la proporción de casos negativos que un modelo reconoce de forma correcta, con respecto al total de instancias

negativas presentes en el conjunto de datos. De modo que, un valor elevado de especificidad indica que el modelo distingue adecuadamente los elementos que no pertenecen a la clase de interés, y reduce la probabilidad de generar clasificaciones incorrectas (Vivanco Toala & Camero Berrones, 2025). En la segmentación de imágenes, la especificidad complementa el análisis del desempeño al reflejar la capacidad del modelo para la disminución de los falsos positivos, evitando que regiones que no corresponden a zonas de acceso terrestre sean identificadas como tales.

Su fórmula es:

$$Especificidad = \frac{VN}{VP + FN}$$

Evaluación y análisis del desempeño de los modelos

Con el propósito de comparar objetivamente el desempeño de las arquitecturas implementadas, se procede a realizar la evaluación utilizando el conjunto de prueba (*Test*), el cual no fue empleado durante las etapas de entrenamiento ni validación. Se analizarán los modelos desde un enfoque cuantitativo y cualitativo, considerando métricas de segmentación a nivel de píxel y la inspección visual de las máscaras generadas.

Evaluación cuantitativa

Se calcularon las principales métricas de segmentación empleando las predicciones obtenidas por cada modelo y sus respectivas máscaras reales. Para ello se utilizaron IoU, Dice, Precisión, Recall, Especificidad y Accuracy, permitiendo evaluar de manera integral la calidad de la segmentación realizada por cada arquitectura. Esto se visualiza en la Tabla 8.

Tabla 8

Métricas de desempeño (%)

Métrica	U-Net	LinkNet	DeepLabV3+
IoU (%)	78.07	80.67	**84.35**

Dice (%)	87.68	89.30	**91.51**
Precision (%)	89.20	89.42	**93.97**
Recall (%)	86.21	**89.18**	89.17
Specificity (%)	99.46	99.45	**99.70**
Accuracy (%)	98.80	98.94	**99.18**

Nota. Elaboración propia.

Como se observa en la Tabla 8, las tres arquitecturas alcanzaron valores superiores al 78 % en IoU y al 87 % en la métrica de Dice, indicando que los tres modelos tienen una alta capacidad para segmentar las vías terrestres accesibles presentes en las imágenes del conjunto de prueba. Sin embargo, se observa que DeepLabV3+ obtuvo el mejor desempeño en la mayoría de las métricas evaluadas, alcanzando un IoU de 84.35 %, un Dice de 91.51 %, una precisión de 93.97 %, una especificidad de 99.70 % y una exactitud de 99.18 %. Por su parte, LinkNet presentó un rendimiento competitivo, con valores cercanos a los obtenidos por DeepLabV3+, mientras que U-Net obtuvo resultados ligeramente inferiores, aunque mantiene un comportamiento equilibrado en todas las métricas.

En cuanto a la precisión, DeepLabV3+ obtuvo el mayor valor (93.97 %), seguido por LinkNet (89.42 %) y U-Net (89.20 %). Este resultado es relevante ya que una mayor precisión se relaciona con una menor cantidad de falsos positivos, aspecto que será analizado con mayor detalle mediante la matriz de confusión y el conteo de píxeles correctamente e incorrectamente clasificados.

Respecto al Recall, las diferencias entre los modelos fueron reducidas, con valores cercanos al 89 %. LinkNet obtuvo el valor más alto (89.18 %), seguido muy de cerca por DeepLabV3+ (89.17 %), mientras que U-Net alcanzó 86.21 %. Finalmente, la especificidad y la

exactitud superaron el 99 % en las tres arquitecturas, reflejando un comportamiento consistente durante la clasificación de los píxeles del conjunto de prueba.

- **Tabla VP, VN, FP, FN**

Complementariamente se analizaron los valores absolutos de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos obtenidos por cada modelo tal como se muestra en la Tabla 9.

Tabla 9

Estadísticos de clasificación a nivel de píxel

	U-Net	**LinkNet**	**DeepLabV3+**
VP	422,439	**436,953**	436,928
VN	9,354,822	9,354,257	**9,377,933**
FP	51,130	51,695	**28,019**
FN	67,545	**53,031**	53,056
Total píxeles	9,895,936	9,895,936	9,895,936

Nota. Elaboración propia.

La Tabla 9 presenta el número total de píxeles que han sido clasificados como verdaderos positivos (VP), verdaderos negativos (VN), falsos positivos (FP) y falsos negativos (FN) para las tres arquitecturas que fueron evaluadas en el presente trabajo. El conjunto de prueba estuvo conformado por 151 imágenes de 256 × 256 píxeles, lo que corresponde a un total de 9 895 936 píxeles utilizados durante la evaluación.

Se observa que la mayor parte de los píxeles pertenece a la clase de verdaderos negativos, lo que indica que las regiones no correspondientes a vías accesibles ocupan una proporción considerable dentro de las imágenes del conjunto de prueba. Este comportamiento

es consistente con las características del escenario post inundación y con la distribución de clases presente en el conjunto de datos.

Un aspecto de especial interés corresponde al análisis de los falsos positivos, ya que este tipo de error implica categorizar como transitable una zona que en realidad no lo es. En escenarios de respuesta ante inundaciones, esta situación podría ocasionar una planificación incorrecta de las rutas de acceso y comprometer tanto el tiempo de respuesta como la seguridad del personal de rescate.

En este sentido, DeepLabV3+ obtuvo la menor cantidad de falsos positivos (28 019 píxeles), frente a los 51 130 de U-Net y 51 697 de LinkNet. Para cuantificar esta diferencia se calculó el porcentaje de reducción mediante la siguiente ecuación:

$$Reducción(\%) = \frac{Valor\ inicial - Valor\ final}{Valor\ inicial} x100$$

Aplicando la ecuación para el caso de U-Net:

$$Reducción(\%) = \frac{51130\ píxeles - 28019\ píxeles}{51130} x100 = 45.20\ \%$$

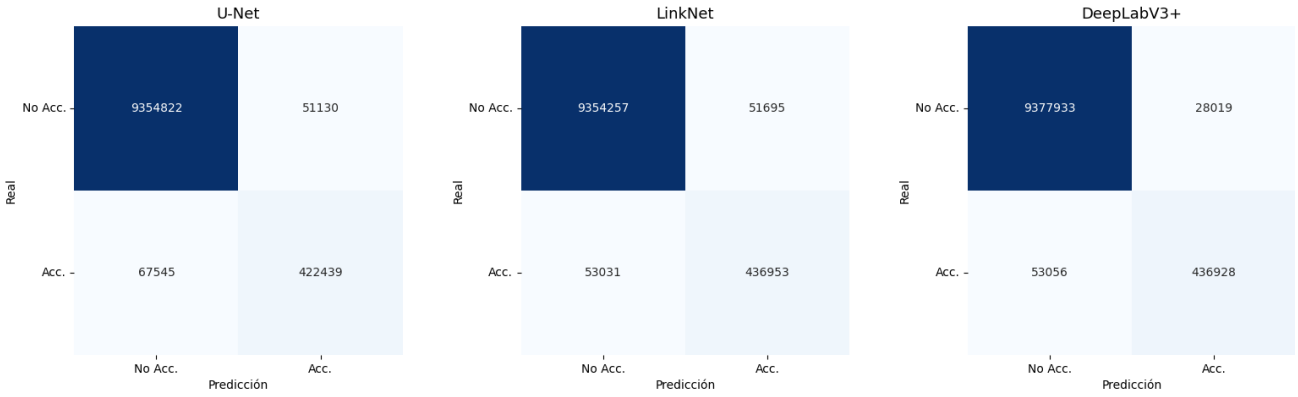
Esto significa que DeepLabV3+ redujo aproximadamente un 45 % los falsos positivos respecto a U-Net y LinkNet, disminuyendo la probabilidad de clasificar zonas no accesibles como vías de acceso terrestre. Este comportamiento es coherente con la mayor precisión obtenida por esta arquitectura y constituye una ventaja importante para aplicaciones orientadas al apoyo de operaciones de rescate.

- **Matrices de confusión**

Con el propósito de complementar el análisis cuantitativo, en la Figura 26 se presentan las matrices de confusión obtenidas para las tres arquitecturas evaluadas. Esta representación permite visualizar de forma gráfica la distribución de las predicciones correctas e incorrectas realizadas por cada modelo.

Figura 26

Matrices de Confusión de los Modelos de Segmentación



Nota. Elaboración propia

Como se aprecia en la Figura 26, en los tres modelos predomina la clasificación correcta de los píxeles, evidenciada por la elevada concentración de verdaderos negativos y verdaderos positivos. De igual manera, la cantidad de falsos positivos y falsos negativos es considerablemente menor, lo que confirma el buen desempeño general de las arquitecturas. Además, se observa que DeepLabV3+ presenta una menor cantidad de falsos positivos respecto a U-Net y LinkNet, resultado que coincide con el análisis presentado en la Tabla 9 y reafirma su mayor capacidad para evitar la clasificación errónea de zonas no accesibles como vías transitables.

Evaluación cualitativa

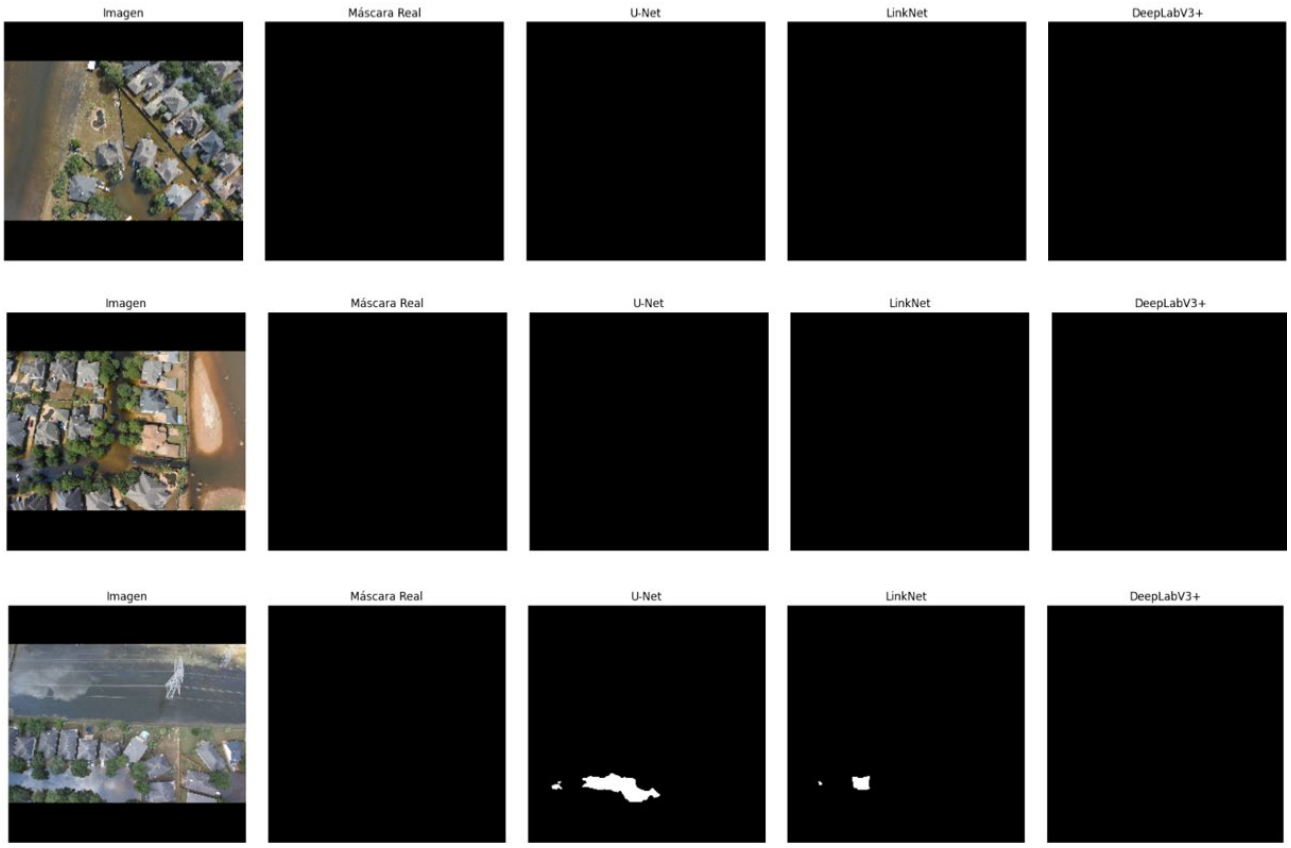
Además del análisis numérico, se realiza una evaluación visual de los mapas binarios obtenidos por cada arquitectura con el fin de observar la capacidad de reconstrucción de las vías accesibles en escenarios de inundación. Para ello, se seleccionaron imágenes representativas del conjunto de prueba con el fin de analizar el comportamiento de los modelos tanto en escenas sin presencia de vías accesibles como en imágenes donde sí existen zonas transitables.

- **Imágenes sin presencia de vías de acceso terrestre**

En esta primera evaluación se analizaron tres imágenes del conjunto de prueba que no contienen píxeles pertenecientes a la clase vía de acceso terrestre. En la Figura 27 se observan las imágenes para este caso con sus respectivas predicciones.

Figura 27

Predicciones de imágenes sin presencia de vías de acceso terrestre



Nota. Elaboración propia.

De la Figura 27 se visualiza que en las dos primeras imágenes las tres arquitecturas generaron correctamente una máscara completamente negra, evidenciando que fueron capaces de identificar la ausencia total de caminos transitables.

Sin embargo, en la última imagen se observa un comportamiento diferente. La presencia de reflejos solares y variaciones de iluminación en el agua provocó que U-Net y

LinkNet segmentaran erróneamente fragmentos inundados como caminos de acceso mientras que DeepLabV3+ produjo un mapa de segmentación completamente negro, coincidiendo con la máscara de referencia de manera que evitó este tipo de errores.

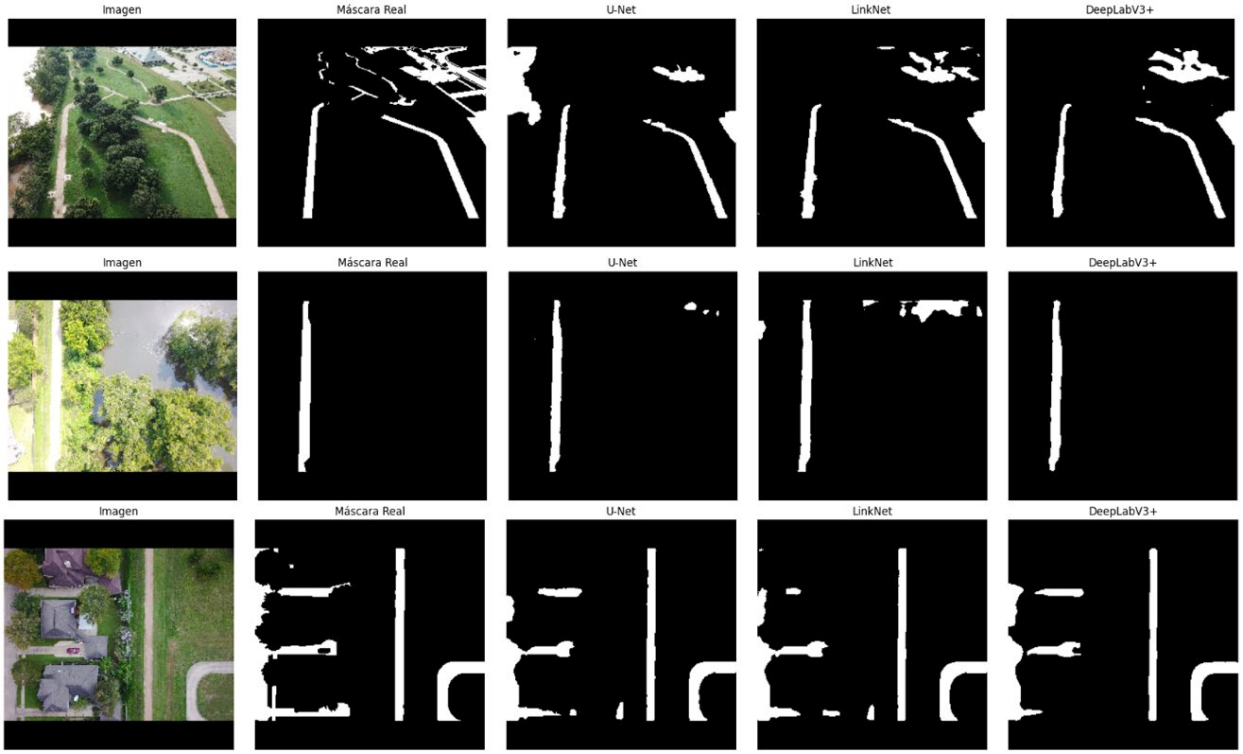
Este resultado coincide con el análisis cuantitativo presentado anteriormente, donde DeepLabV3+ obtuvo la menor cantidad de falsos positivos, demostrando que puede distinguir mejor entre áreas inundadas y caminos que realmente se pueden utilizar.

- **Imágenes con presencia de vías de acceso terrestre**

Las imágenes con presencia de vías de acceso terrestre se organizaron en dos grupos representativos con el fin de analizar el comportamiento de las arquitecturas en diferentes escenarios.

Figura 28

Predicciones de imágenes con presencia de vías de acceso terrestre

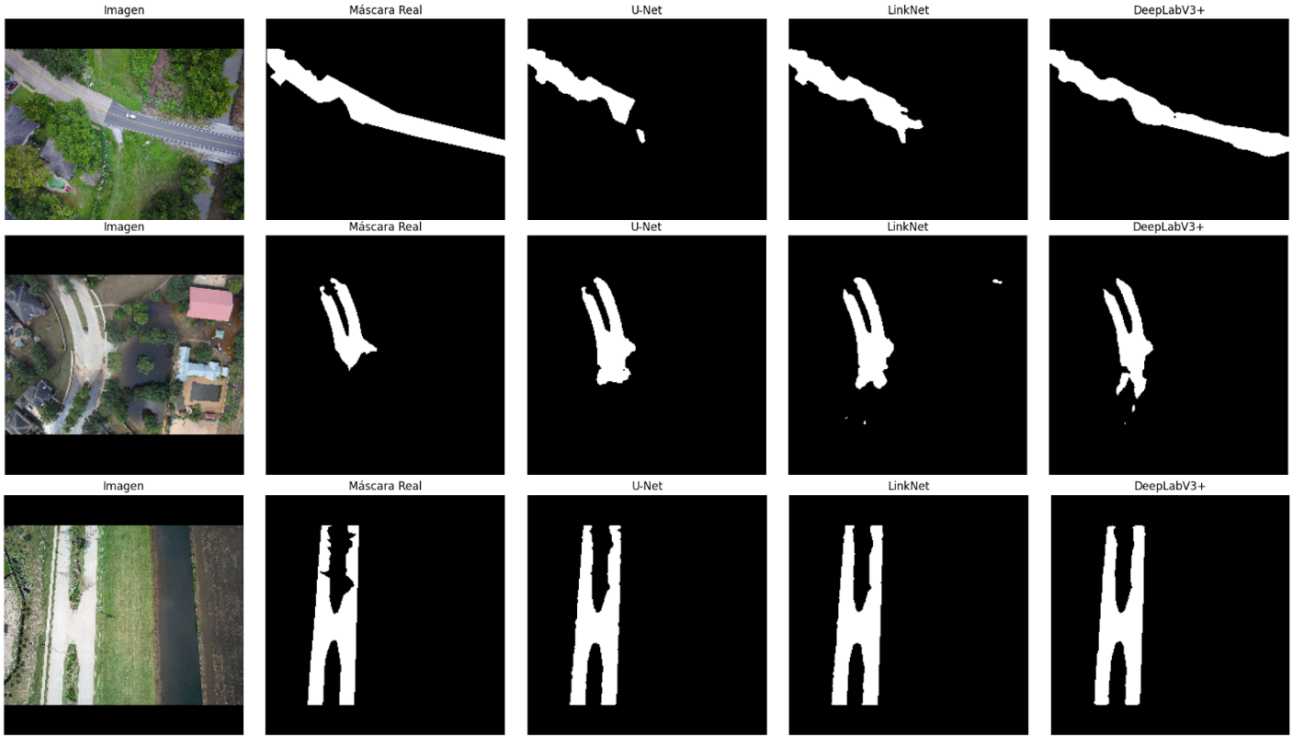


Nota. Elaboración propia.

En la Figura 28 se observa que las tres arquitecturas identifican las vías principales; sin embargo, U-Net y LinkNet presentan regiones adicionales clasificadas erróneamente como vías de acceso terrestre, principalmente sobre superficies inundadas o zonas con reflejos que poseen características visuales similares a una carretera. En contraste, DeepLabV3+ genera segmentaciones más cercanas a la máscara de referencia, reduciendo la aparición de falsos positivos y delimitando con mayor precisión las vías realmente transitables.

Figura 29

Predicciones de imágenes con presencia de vías de acceso terrestre



Nota. Elaboración propia.

Por otro lado, en la Figura 29 se presentan escenas donde las diferencias entre los modelos son menos evidentes. Para la primera imagen, U-Net y LinkNet no logran segmentar completamente la vía de acceso terrestre, generando una región correspondiente a un falso negativo. Por el contrario, DeepLabV3+ consigue detectar la totalidad de la vía, obteniendo una segmentación más completa. Las dos últimas imágenes de la Figura 29 representan ejemplos

en donde los tres modelos producen resultados muy similares. Esto muestra que cuando las características de la vía son fáciles de distinguir, los tres modelos pueden realizar una segmentación satisfactoria.

Sistema final para la detección automática de vías de acceso terrestre

A partir de los resultados obtenidos durante la evaluación, DeepLabV3+ fue seleccionado como la arquitectura definitiva para la implementación del módulo de segmentación, al alcanzar el mejor desempeño global entre los modelos evaluados. Como resultado final del presente trabajo de titulación se desarrolló un sistema capaz de recibir una imagen aérea como entrada y generar de forma automática la máscara binaria correspondiente a las vías de acceso terrestre detectadas en los escenarios post inundación. Para mantener la validez del modelo, la aplicación ejecuta exactamente los mismos pasos de acondicionamiento de imagen usados en el entrenamiento de forma que así favorece la reproducibilidad de los resultados obtenidos.

A diferencia del proceso de entrenamiento en el cual fue necesario preprocesar y almacenar el conjunto completo de imágenes en arreglos NumPy para optimizar la lectura durante las etapas de entrenamiento y evaluación, este módulo final procesa una imagen de entrada de manera individual de manera que se asemeja más a un escenario de utilización de aplicación real. Este enfoque permite analizar cada fotografía capturada por un dron a medida que es adquirida, de forma que facilita su incorporación dentro de un flujo continuo de monitoreo para la generación progresiva de mapas de vías accesibles posteriores a un evento de inundación. Por tal motivo, el preprocesamiento se realiza de manera individual antes de la inferencia, eliminando la necesidad de generar archivos .npy tal como se realizó para el entrenamiento de las arquitecturas.

Para interpretar los resultados con mayor facilidad, además de la máscara binaria segmentada, el sistema genera una imagen de superposición (*overlay*) sobre la fotografía original. Esta representación es de suma importancia ya que permite identificar visualmente las

zonas clasificadas como vías de acceso terrestre, por lo que es una herramienta de apoyo para la validación de la segmentación y la interpretación de los resultados por parte del usuario.

Con esta implementación se materializa el alcance planteado para el presente trabajo de titulación, ya que una imagen aérea capturada por un UAV es adquirida por el sistema, preprocesada y, posteriormente, analizada por el modelo de segmentación basado en redes neuronales convolucionales. Como resultado, se obtiene de manera automática un mapa binario que identifica las vías de acceso terrestre detectadas. La Figura 30 muestra los resultados del módulo desarrollado.

Figura 30

Resultados de implementación del módulo final de detección



Nota. Elaboración propia.

En la implementación del módulo final se empleó un redimensionamiento directo de la imagen de entrada a 256 × 256 píxeles. A diferencia de la etapa de entrenamiento, en la cual se aplicó *padding* para preservar la relación de aspecto durante la construcción del conjunto de datos, en este sistema final se optó por un preprocesamiento más simple orientado a la

inferencia sobre imágenes individuales. Esta decisión facilita la visualización del resultado y no modifica las métricas reportadas, ya que dichas métricas fueron obtenidas utilizando el procedimiento de preprocesamiento definido para la evaluación de los modelos.

Discusión de resultados

Gracias a los resultados obtenidos se logró analizar el comportamiento de las tres arquitecturas de segmentación que han sido empleadas en el presente trabajo de titulación.

DeepLabV3+ obtuvo el mejor desempeño debido a que combina dos características fundamentales para este tipo de imágenes. La inclusión del módulo ASPP resultó clave para analizar la escena a distintas escalas espaciales de manera que permitió identificar tramos de carretera cubiertos parcialmente por inundaciones o vegetación. Asimismo, la reconexión de mapas de características desde la fase inicial del codificador hacia el decodificador ayudó a generar la reconstrucción de los contornos de las vías con mayor exactitud, reduciendo significativamente la aparición de falsos positivos en las inferencias.

Además de las ventajas propias de la arquitectura, el mejor desempeño de DeepLabV3+ también se relaciona con las características particulares de las imágenes empleadas en esta investigación. El conjunto FloodNet contiene escenas aéreas adquiridas mediante UAV en las que las vías de acceso representan únicamente una pequeña fracción de la imagen y aparecen rodeadas por agua, vegetación, edificaciones y zonas parcialmente inundadas. Estas condiciones incrementan la complejidad de la segmentación, ya que existen regiones con características visuales similares a una vía, tal como pueden ser reflejos sobre el agua o superficies con alta iluminación.

Asimismo, el decodificador de DeepLabV3+ permitió recuperar detalles espaciales que suelen perderse durante el proceso de extracción de características profundas. Esto logró evidenciarse sobre todo en aquellas imágenes donde las vías presentaban cambios de dirección, intersecciones o segmentos parcialmente cubiertos por vegetación, de forma que se obtuvo contornos más continuos, así como una reducción visible de falsas detecciones de vías

respecto a U-Net y LinkNet. Esto se pudo visualizar en la evaluación cualitativa en donde DeepLabV3+ presentó un mapa de segmentación binario más limpio en imágenes con presencia de reflejos sobre superficies inundadas, además de presentar también una mejor continuidad de las vías en escenas con presencia de objetos que obstruían los caminos.

Por otro lado, LinkNet presentó un desempeño competitivo debido al uso de conexiones residuales y conexiones de suma entre *encoder* y *decoder* lo que favorece una propagación eficiente de la información y una reconstrucción adecuada de las regiones segmentadas. Sin embargo, al no incorporar un mecanismo explícito para capturar el contexto a diferentes escalas, como el módulo ASPP de DeepLabV3+, esta arquitectura mostró una mayor tendencia a generar falsas detecciones sobre superficies inundadas con características visuales similares a una vía.

Aunque U-Net obtuvo las métricas más bajas entre las arquitecturas evaluadas, los resultados evidencian que continúa siendo una arquitectura sólida para realizar tareas de segmentación semántica tras haber alcanzado un IoU del 78.07%. Su diseño basado en conexiones de salto funcionó bien para reconstruir la mayoría de las vías no inundadas, en especial cuando dichas vías tenían formas claras. Sin embargo, tuvo problemas en escenarios con masas de agua o mucha complejidad visual, generando muchos falsos positivos y perdiendo continuidad.

Por otro lado, desde el punto de vista computacional, las tres arquitecturas pudieron entrenarse empleando la versión gratuita de Google Colab, utilizando una GPU NVIDIA Tesla T4 con aproximadamente 15 GB de memoria. Para ajustarse a estos recursos fue necesario redimensionar las imágenes a una resolución de 256 × 256 píxeles y emplear un tamaño de lote de ocho imágenes. Aunque DeepLabV3+ presentó una arquitectura más compleja que U-Net, su tiempo promedio de inferencia sobre el conjunto de prueba fue de 0.0151 segundos por imagen, manteniéndose adecuado para aplicaciones que requieren una respuesta rápida. Gracias a estos resultados se demuestra que es posible implementar modelos de

segmentación semántica con un buen desempeño utilizando recursos computacionales gratuitos y ampliamente accesibles, además de favorecer la reproducibilidad de la metodología propuesta y su futura integración en sistemas de apoyo para la gestión de emergencias.

Finalmente, más allá de la comparación entre arquitecturas, uno de los principales aportes de esta investigación fue la implementación de un módulo funcional de segmentación basado en DeepLabV3+, el cual integra el flujo completo desarrollado durante el trabajo, lo que involucra: preprocesamiento de la imagen, inferencia mediante el modelo entrenado y generación automática de la máscara binaria de vías de acceso terrestre junto con su superposición sobre la imagen original.

Limitaciones del sistema

Si bien el sistema desarrollado obtuvo resultados satisfactorios en la segmentación de vías terrestres accesibles en escenarios de post inundación, hay algunas limitaciones. Estas limitaciones se relacionan tanto al conjunto de datos como a las condiciones propias de las imágenes y del entorno de desarrollo, influyendo en el desempeño del modelo. Las limitaciones se mencionan a continuación:

- Limitación por dependencia del conjunto de datos, debido a que el modelo fue entrenado y evaluado exclusivamente con imágenes del dataset FloodNet, capturadas mediante UAV en escenarios post inundación. Aunque este conjunto presenta una gran variedad de escenas, las imágenes mantienen características similares en cuanto a la altura de captura y condiciones propias de este desastre. En consecuencia, el desempeño del modelo frente a imágenes obtenidas con diferentes plataformas, resoluciones, regiones geográficas o en otros tipos de desastres naturales deberá ser validado antes de su aplicación en escenarios distintos.
- Limitación por presencia de oclusiones y elementos que dificultan la identificación completa de las vías terrestres, como vegetación, edificaciones,

vehículos o sombras proyectadas sobre la superficie. Esto puede hacer que los resultados predichos no sean exactos en comparación a las máscaras reales, especialmente cuando el camino está parcialmente cubierto o no se ve muy bien.

- Limitación en cuanto al recurso computacional, ya que el desarrollo y entrenamiento de las arquitecturas se realizó utilizando la versión gratuita de Google Colab, la cual dispone de recursos computacionales limitados. Por esto, fue necesario trabajar con imágenes de 256 × 256 píxeles y un *batch size* de 8 durante el entrenamiento. Estas decisiones permitieron ejecutar correctamente los experimentos, aunque sería interesante investigar si se pueden usar imágenes de mayor resolución con computadoras más potentes.

Capítulo V: Conclusiones y recomendaciones

En este capítulo se exponen las principales conclusiones derivadas del desarrollo del presente trabajo de titulación. Además, se presentan recomendaciones y trabajos futuros que contribuyan a fortalecer y ampliar el alcance de esta investigación.

Conclusiones

- Se realizó un estudio sobre técnicas de segmentación semántica aplicadas a escenarios de desastres naturales, identificando a U-Net, LinkNet y DeepLabV3+ como arquitecturas ampliamente utilizadas para la segmentación de imágenes. Asimismo, se analizaron conceptos fundamentales como convolución, *pooling*, funciones de activación*, encoder-decoder, skip connections y backbones* preentrenados, proporcionando la base teórica para el desarrollo de la investigación.
- Se diseñó un esquema de segmentación binaria orientado a la identificación de vías de acceso terrestre mediante la adaptación del conjunto de datos *FloodNet*.

Para ello se generaron máscaras binarias y se realizaron procesos de corrección manual y preprocesamiento de las respectivas imágenes, de forma que se obtuvo un conjunto de datos adecuado para el entrenamiento y evaluación de las arquitecturas implementadas.

- Se implementaron satisfactoriamente las arquitecturas U-Net, LinkNet y DeepLabV3+ empleando TensorFlow/Keras e incorporando de funciones de pérdida, callbacks y ajustes de hiperparámetros que permitieron optimizar el proceso de entrenamiento. Gracias a los resultados se evidenció que cada arquitectura presentó un comportamiento diferente frente al problema de segmentación ya que, U-Net requirió un mayor proceso de optimización para mejorar su capacidad de generalización, LinkNet alcanzó un desempeño competitivo con una estructura eficiente, mientras que DeepLabV3+ obtuvo la mejor capacidad para segmentar las vías de acceso terrestre ante escenarios de inundación, de forma que se demuestra las ventajas del uso de un *backbone preentrenado* y del módulo ASPP para la extracción de características multiescala.
- La evaluación cuantitativa y cualitativa permitió verificar la precisión y robustez de las arquitecturas implementadas para la detección de vías de acceso terrestre. DeepLabV3+ obtuvo los mejores resultados, alcanzando un IoU de 84.35 % y un coeficiente Dice de 91.51 %, además de reducir aproximadamente un 45 % los falsos positivos respecto a U-Net y LinkNet. Estos resultados permitieron seleccionar esta arquitectura para la implementación del módulo final de segmentación, el cual recibe una imagen aérea como entrada y genera automáticamente el mapa binario de las vías de acceso terrestre detectadas. De esta manera, se cumple el objetivo planteado de desarrollar un módulo capaz de

apoyar la identificación automática de rutas potencialmente transitables en escenarios post inundación.

Recomendaciones

- Se recomienda mantener un proceso cuidadoso de preparación del conjunto de datos, de forma que es imperativo verificar la correcta correspondencia entre imágenes y máscaras. La precisión de las máscaras de referencia es muy importante para el desempeño de los modelos de segmentación.
- Se recomienda mantener un buen equilibrio entre la resolución de entrada, el tamaño del lote y la capacidad del hardware disponible. Esta configuración permite entrenar los modelos de forma estable utilizando los recursos computacionales disponibles para evitar una sobrecarga de la memoria.

Trabajos futuros

Incorporar *backbones* preentrenados en arquitecturas como U-Net y LinkNet, con el fin de aprovechar el conocimiento previamente adquirido durante el entrenamiento sobre grandes conjuntos de imágenes. Esto podría ayudar a que los modelos converjan más rápido y mejoren su desempeño en cuanto a segmentación.

Además, como trabajo futuro se propone integrar el modelo desarrollado dentro de una plataforma de apoyo que se encuentre orientada a la gestión de emergencias, de forma que reciba imágenes capturadas por UAV para procesarlas automáticamente y así identificar vías terrestres accesibles después de un evento de inundación.

Asimismo, el sistema podría integrarse con plataformas de gestión de riesgos y organismos de respuesta ante emergencias, como el ECU 911, permitiendo generar información geoespacial en tiempo casi real. Esto sería muy útil para apoyar la planificación de rutas de acceso así como la coordinación de equipos de rescate y la toma rápida de decisiones durante eventos de desastre.

Finalmente, se propone como línea futura de investigación la evaluación del desempeño del sistema utilizando imágenes obtenidas en diferentes tipos de desastres naturales como deslizamientos de tierra o terremotos, todo esto con el objetivo de analizar la capacidad de generalización del modelo y ampliar su aplicación a distintos escenarios donde se requiera una gestión adecuada y oportuna ante riesgos.

Referencias:

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Brain, G. (2016). *TensorFlow: A system for large-scale machine learning*. https://github.com/

Akash. (2013). *Normalización RGB* . https://akash0x53.github.io/blog/2013/04/29/RGB-Normalization/

Aldea Blasco, I. (2019). *Redes Neuronales Convolucionales. Aspectos teóricos y aplicaciones en aprendizaje supervisado.*

AprendeIA. (2019). *Matriz de Confusión* . https://aprendeia.com/2019/05/17/matriz-de-confusion-machine-learning/

Arozarena Villar, A. (n.d.). *TELEDETECCIÓN Y SISTEMAS DE TRATAMIENTO DIGITAL DE IMÁGENES* . Retrieved July 6, 2026, from https://moodle.upm.es/en-abierto/pluginfile.php/46216/mod_label/intro/microsoft-word-teledeteccion_y_sist_tratamiento_digital_imagenes.pdf

Astudillo, V., & Revelo, D. (2021). *Apoyo al diagnóstico de neumonía y detección de opacidades pulmonares usando segmentación de instancias semánticas en imágenes de rayos X de tórax*. https://doi.org/10.14482/inde.39.2.621.367

Bernal, C. A. (2010). *Metodología de la investigación* (TERCERA EDICIÓN E-B…). www.FreeLibros.me

Borja, R., Monleón, A., & Rodellar, J. (2020). *Estandarización de Métricas de Rendimiento para Clasificadores Machine y Deep Learning*. https://www.researchgate.net/publication/339943922_Estandarizacion_de_Metricas_de_Rendimiento_para_Clasificadores_Machine_y_Deep_Learning

Bradski, Gary. (2000). The OpenCV library - ProQuest. *Dr. Dobb's Journal of Software Tools*. https://www.proquest.com/docview/202684726?sourcetype=Trade%20Journals

Cando Garces, S. K. (2018). *Desarrollo de algoritmos de inteligencia artificial para la segmentación y clasificación de hiperintensidades en sustancia blanca cerebral obtenidas de imágenes de Resonancia Magnética.* .

Chaurasia, A., & Culurciello, E. (2017). LinkNet: Exploiting encoder representations for efficient semantic segmentation. *2017 IEEE Visual Communications and Image Processing (VCIP)*, *2018-January*, 1–4. https://doi.org/10.1109/VCIP.2017.8305148

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). *Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation*. https://github.com/tensorflow/models/tree/master/

Clark, A. (2024). *Pillow (PIL Fork) 12.3.0 documentation*. https://pillow.readthedocs.io/en/stable/

Codificando Bits. (2019). *Padding, strides, max-pooling y stacking en las Redes Convolucionales*. https://codificandobits.com/blog/padding-strides-maxpooling-stacking-redes-convolucionales/

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2021). *ImageNet: A Large-Scale Hierarchical Image Database*. http://www.image-net.org.

Díaz Olariaga, O. (2024). Vehículos aéreos no tripulados en la gestión de emergencia por desastre. *Boletín Geográfico*, 1–21.

ECU 911. (2024). *El ECU 911 coordinó atención ante emergencias por lluvias en Ibarra y Antonio Ante*. https://www.ecu911.gob.ec/el-ecu-911-coordino-atencion-ante-emergencias-por-lluvias-en-ibarra-y-antonio-ante/

Fawakherji, M., Blay, J., Anokye, M., Hashemi-Beni, L., & Dorton, J. (2025). DeepFlood for Inundated Vegetation High-Resolution Dataset for Accurate Flood Mapping and Segmentation. *Scientific Data* , *12*(1). https://doi.org/10.1038/s41597-025-04554-3

Fronzetti, N. (2019). *Predictive Neural Network Applications for Insurance Processes*.

García Vega, J. (2024). *Estudio e implementación de redes neuronales convolucionales para la segmentación de imágenes*. https://gredos.usal.es/bitstream/handle/10366/163859/memoria.pdf?sequence=1

García Villalba, L. J., & Armas Vega, E. A. (2021). *Algoritmo de Pre-Procesamiento de Imágenes para la Estimación de la Edad*.

GeeksforGeeks. (2025). *Tensorflow Callbacks* . https://www.geeksforgeeks.org/deep-learning/tf-keras-callbacks-callback-tensorflow-callbacks/

Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2019). Fundamental concepts of convolutional neural network. *Intelligent Systems Reference Library*, *172*, 519–567. https://doi.org/10.1007/978-3-030-32644-9_36

Harris, C. R., Jarrod Millman, K., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Fernández del Río, J., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*, 357. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Jadon, S. (2020). *A survey of loss functions for semantic segmentation*. https://github.com/shruti-jadon/

Ji, Y., Zhang, H., Zhang, Z., & Liu, M. (2021). CNN-based encoder-decoder networks for salient object detection: A comprehensive review and recent advances. *Information Sciences*, *546*, 835–857. https://doi.org/10.1016/J.INS.2020.09.003

Kage, P., & Rothenberger, J. C. (2025). *A Review of Pseudo-Labeling for Computer Vision*.

Kainat. (2023). *Codificador-Decodificador: Extracción de Características*. https://medium.com/@kitkat73275/encoder-decoder-features-extraction-fbccfc31c047

Kimm, H., Paik, I., & Kimm, H. (2021). Performance Comparision of TPU, GPU, CPU on Google Colaboratory over Distributed Deep Learning. *Proceedings - 2021 IEEE 14th International Symposium on Embedded Multicore/Many-Core Systems-on-Chip, MCSoC 2021*, 312–319. https://doi.org/10.1109/MCSoC51149.2021.00053

Kingma, D. P., & Lei Ba, J. (2017). *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION*.

Lata, B. (2022). *Architecture of Unet skip connection*. https://www.researchgate.net/figure/Architecture-of-Unet-skip-connection_fig1_376158645

Li, G., Liu, Y., Shen, D., & Wang, B. (2024). Automatic pixel-level bridge crack detection using learning context flux field with convolutional feature fusion. *Journal of Civil Structural Health Monitoring*, *14*(5), 1155–1171. https://doi.org/10.1007/S13349-024-00775-Z

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(12), 6999–7019. https://doi.org/10.1109/TNNLS.2021.3084827

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Lawrence Zitnick, C., & Irvine, U. (2014). Microsoft COCO: Common Objects in Context. *In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13 (Pp. 740-755). Springer International Publishing.*

Matamoros Pinto, K. A. (2025). *Identificación de patrones en imágenes preprocesadas para la detección de enfermedades y comportamientos anómalos en animales mediante backbones de redes neuronales*.

Miguélez, Á. (2022). *Aprendizaje profundo para segmentación panóptica de escenas robusta a adaptación de dominio*.

Morales, B., Garcia-Pedrero, A., Lizama, E., Lillo-Saavedra, M., Gonzalo-Martín, C., Chen, N., & Somos-Valenzuela, M. (2022). Patagonian Andes Landslides Inventory: The Deep Learning's Way to Their Automatic Detection. *Remote Sensing*, *14*(18). https://doi.org/10.3390/RS14184622

Muñoz Gómez, A. C. (2025). *Índice de vulnerabilidad vial por inundación aplicado a una zona metropolitana considerando la accesibilidad basada en oportunidades y datos viales*.

Pan, S. J., & Yang, Q. (2009). *A Survey on Transfer Learning*. https://doi.org/10.1109/TKDE.2009.191

Patiño Duarte, D. P., Vargas Rodriguez, J., & Rodriguez Vaca, M. (2026). *Estrategias para garantizar una respuesta rápida y eficaz ante emergencias mediante un análisis de vulnerabilidad y amenazas en UNIMINUTO CU Cúcuta*. 0–63.

Python Software Foundation. (2025a). *json — JSON encoder and decoder — Python 3.14.6 documentation*. Python Documentation. https://docs.python.org/3/library/json.html

Python Software Foundation. (2025b). *os — Miscellaneous operating system interfaces — Python 3.14.6 documentation*. Python Documentation. https://docs.python.org/3/library/os.html

Python Software Foundation. (2025c). *tarfile — Read and write tar archive files — Python 3.14.6 documentation*. Python Documentation. https://docs.python.org/3/library/tarfile.html

Rahnemoonfar, M., Chowdhury, T., Sarkar, A., Varshney, D., Yari, M., & Murphy, R. (2020). *FloodNet: A High Resolution Aerial Imagery Dataset for Post Flood Scene Understanding*.

Redondo Carpio, E. (2024). *Aplicación de OIF (Optimum Index Factor) en Clasificación de Imágenes de Teledetección.* https://oa.upm.es/82474/1/TFG_ESTER_REDONDO_CARPIO.pdf

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

Sánchez Álvarez, E. (2024). *Proyecto urbano para mejorar la accesibilidad urbana ante el riesgo de inundación*.

Sani, J. C., Tierra Criollo, A. R., & Robayo Nieto, A. A. (2015). *Vehículos aéreos no tripulados - UAV para la elaboración de cartografías escalas grandes referidas al marco de referencia SIRGAS-ECUADOR*.

Shorten, C., & Khoshgoftaar, T. M. (2019). *A survey on Image Data Augmentation for Deep Learning*. https://doi.org/10.1186/s40537-019-0197-0

Si, Y., Gong, D., Guo, Y., Zhu, X., Huang, Q., Evans, J., He, S., & Sun, Y. (2021). An Advanced Spectral–Spatial Classification Framework for Hyperspectral Imagery Based on DeepLab v3+. *Applied Sciences 2021, Vol. 11, Page 5703*, *11*(12), 5703. https://doi.org/10.3390/APP11125703

Skalski, P. (2019). *Make Sense*. https://www.makesense.ai/

Supervisely. (2024). *What's Supervisely | Supervisely*. Supervisely Documentation. https://docs.supervisely.com/

Swapna. (2020, August). *Convolutional Neural Network* . Developers Breach. https://developersbreach.com/convolution-neural-network-deep-learning/

Tomar, N. (2021). *DeepLabV3+ with ResNet50 as Encoder in TensorFlow 2.5 | DeepLabV3Plus | UNET - YouTube*. https://www.youtube.com/watch?v=Ik85ry-Rf1g

Vivanco Toala, D. M., & Camero Berrones, R. G. (2025). Determinación de un Modelo mediante Aprendizaje Automático para obtener la Idoneidad del Personal Naval Profesional. *Ciencia Latina Revista Científica Multidisciplinar*, *9*(4), 8653–8676. https://doi.org/10.37811/cl_rcm.v9i4.19442

Zou, K. H., Warfield, S. K., Bharatha, A., Tempany, C. M. C., Kaus, M. R., Haker, S. J., Wells Iii, W. M., Jolesz, F. A., Kikinis, R., & St, F. (2004). Statistical Validation of Image Segmentation Quality Based on a Spatial Overlap Index 1 : Scientific Reports. *Acad Radiol*, *11*(2), 178–189. http://www.slicer.org

Apéndices

En el siguiente enlace se encuentra el código fuente correspondiente al módulo desarrollado en este trabajo. Se incluye el preprocesamiento del conjunto de datos, la implementación y entrenamiento de las arquitecturas U-Net, LinkNet y DeepLabV3+, los modelos entrenados, el código empleado para la evaluación final de los resultados y el módulo final de detección implementado.

https://github.com/karenlara1108-prog/LARA-KAREN---TRABAJO-DE-TITULACION-