UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS

SEGMENTACIÓN AUTOMATIZADA DE LAGUNAS EN LA CORDILLERA DEL VILCANOTA, REGIÓN CUSCO, UN ENFOQUE BASADO EN IMÁGENES SATELITALES Y APRENDIZAJE PROFUNDO

PRESENTADOR POR:
Br. WILLIAM ISAAC PEREZ TORRES

PARA OPTAR AL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

ASESOR:
Dr. FACUNDO PALOMINO QUISPE

CUSCO - PERÚ

2024

INFORME DE ORIGINALIDAD

(Aprobado por Resolución Nro.CU-303-2020-UNSAAC)

El que suscribe, **Asesor** del trabajo deinvestigación/tesis titulada:.....SEGMENTACIÓN AUTOMATIZADA DE LAGUNAS EN LA CORDILLERA DEL VILCANOTA, REGIÓN CUSCO, UN ENFOQUE BASADO EN IMÁGENES SATELITALES Y APRENDIZAJE PROFUNDO.....

presentado por: WILLIAM ISAAC PEREZ TORRES con DNI Nro.: 70577302 presentado por: ..... con DNI Nro.: ..... para optar el título profesional/grado académico de INGENIERO ELECTRÓNICO

.....

Informo que el trabajo de investigación ha sido sometido a revisión por 02 veces, mediante el Software Antiplagio, conforme al Art. 6° del ***Reglamento para Uso de Sistema Antiplagio de la UNSAAC*** y de la evaluación de originalidad se tiene un porcentaje de 9 %.

Evaluación y acciones del reporte de coincidencia para trabajos de investigación conducentes a grado académico o título profesional, tesis

Porcentaje	Evaluación y Acciones	Marque con una (X)
Del 1 al 10%	No se considera plagio.	X
Del 11 al 30 %	Devolver al usuario para las correcciones.	
Mayor a 31%	El responsable de la revisión del documento emite un informe al inmediato jerárquico, quien a su vez eleva el informe a la autoridad académica para que tome las acciones correspondientes. Sin perjuicio de las sanciones administrativas que correspondan de acuerdo a Ley.	

Por tanto, en mi condición de asesor, firmo el presente informe en señal de conformidad y **adjunto** la primera página del reporte del Sistema Antiplagio.

Cusco, 07 de enero de 20 25

.....
Firma

Post firma FACUNDO PALOMINO QUISPE

Nro. de DNI 00435194

ORCID del Asesor 0000-0002-5947-6682

Se adjunta:

1. Reporte generado por el Sistema Antiplagio.
2. Enlace del Reporte Generado por el Sistema Antiplagio: **oid:** 27259:419515643

William Perez Torres

perez_torres_final.pdf

Universidad Nacional San Antonio Abad del Cusco

Detalles del documento

Identificador de la entrega
trn:oid:::27259:419515643

Fecha de entrega
6 ene 2025, 4:37 p.m. GMT-5

Fecha de descarga
6 ene 2025, 5:31 p.m. GMT-5

Nombre de archivo
perez_torres_final.pdf

Tamaño de archivo
13.9 MB

200 Páginas

51,615 Palabras

270,985 Caracteres

9% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto citado
- Texto mencionado
- Coincidencias menores (menos de 8 palabras)

Exclusiones

- N.º de coincidencias excluidas

Fuentes principales

8% Fuentes de Internet
3% Publicaciones
8% Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alertas de integridad para revisión

No se han detectado manipulaciones de texto sospechosas.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

Dedicatoria

A mi familia, por su incondicional apoyo
y todas las bondades que me brinda.

Agradecimiento

A Dios.

A mis padres, William Pérez y Fanny Torres, quienes me han guiado hasta este momento, permitiéndome culminar este proceso de mi vida. A mis hermanos: Isaac, Carlos y Teresa.

A todos los docentes que compartieron su conocimiento y experiencia conmigo, haciendo posible la realización de este trabajo. Especialmente, al Dr. Ing. Facundo Palomino y a la Dra. Ing. Ana Beatriz Álvarez, por su asesoramiento, supervisión y confianza. Al laboratorio LIECAR, que me brindó el espacio y orientación para la consecución de esta investigación.

A los amigos que hice en el camino.

Índice general

Índice de Tablas

Índice de figuras

Resumen

El monitoreo de las lagunas en la cordillera del Vilcanota es fundamental para entender sus dinámicas y gestionar este recurso en riesgo. Las imágenes satelitales han facilitado el estudio remoto de estos cuerpos de agua, pero el análisis convencional presenta limitaciones, como el extenso trabajo manual requerido para abarcar grandes áreas durante largos períodos. Para abordar estos problemas, esta investigación propone un método basado en aprendizaje profundo para la segmentación automática de lagunas en la cordillera del Vilcanota a partir de imágenes satelitales. Se desarrolló un conjunto de datos específico para esta zona de estudio. El modelo propuesto, UNetFFT, optimiza el modelo de segmentación semántica UNet mediante el módulo Fourier Combination, diseñado para aplicar filtros de parámetros aprendibles en el dominio de la frecuencia y mejorar la detección de bordes y lagunas pequeñas. Los resultados experimentales muestran que UNetFFT obtuvo un 85.94 % en la métrica MIoU (*Mean Intersection over Union*), 81.60 % en *F1-Score*, y 99.67 % en *Pixel Accuracy*, superando a modelos de comparación (FCN, Linknet, PSPNet y UNet). La prueba estadística de Wilcoxon indica que la mejora de UNetFFT frente a UNet es estadístsicamente significativa. Finalmente, al comparar la estimación de áreas de UNetFFT con datos del Instituto Nacional de Investigación en Glaciares y Ecosistemas de Montaña (INAIGEM), no se hallaron diferencias estadísticamente significativas entre ambas mediciones, lo que comprueba la efectividad de UNetFFT para realizar segmentaciones y estimaciones de área precisas.

Palabras clave: Teledetección, Lagunas, Vilcanota, Aprendizaje Profundo, FFT, UNet.

Introducción

La cordillera del Vilcanota, ubicada en los Andes peruanos, es un ecosistema vital que alberga numerosas lagunas de origen glaciar. Con el cambio climático acelerando el deshielo, el monitoreo de estos cuerpos de agua se vuelve esencial para la gestión ambiental. El seguimiento de estos cuerpos basados en imágenes satelitales a menudo se realiza utilizando métodos convencionales cono los índices espectrales, sin embargo, hacerlo de esta forma requiere mucho trabajo manual. Con el auge del procesamiento basado en aprendizaje profundo, se han planteado diversos modelos que realizan la segmentación de lagunas de forma automática, sin embargo, aún se observan limitaciones para diferenciar los cuerpos de agua de su fondo, así como la detección pobre y borrosa de sus bordes. En esta investigación se propone UNetFFT, una optimización de UNet clásico para abordar las problemáticas mencionadas. La presente tesis se estructura en cinco capítulos.

El Capítulo 1: Generalidades, plantea de manera estructurada la problemática que sustenta este trabajo, así como los objetivos, alcances y limitaciones de la investigación. También se describe la metodología utilizada y los flujos de trabajo propuestos.

En el Capítulo 2: Marco teórico, se describen los principales antecedentes relacionados a la propuesta en esta investigación, tomando en cuenta tanto trabajos nacionales como internacionales. Se plantean las definiciones más importantes de las lagunas, las características que las definen y su importancia. Se describe la zona de estudio. Se desarrollan los conceptos más importantes de las imágenes satelitales, insumo básico para este trabajo. Se explica la tarea de segmentación de imágenes y los enfoques clásicos utilizados en la segmentación de lagunas. Se define aprendizaje profundo y sus tipos, como las redes neuronales artificiales y convolucionales, base fundamental para la propuesta de la tesis. Se desarrollan algunos algoritmos de

procesamiento de imágenes que se utilizan en la investigación. Se desarrolla la teoría de la transformada discreta de Fourier en 1 dimensión, ya que sustenta la aplicación de filtros aplicados en el dominio de la frecuencia. Finalmente, se explican las pruebas estadísticas utilizadas en la investigación.

En el Capítulo 3: Desarrollo e implementación del marco de trabajo, basados en la literatura, se justifican todas las elecciones metodológicas respecto a la adquisición y preparación de datos, las técnicas de procesamiento y la generación final del conjunto de datos para la zona de estudio. Se desarrolla detalladamente el modelo propuesto, UNetFFT, basados en la aplicación de filtros en el dominio de la frecuencia. Se explican los modelos y metodología utilizadas como punto de comparación del modelo y se definen las métricas de evaluación.

En el Capítulo 4: Experimentos y resultados, se definen los experimentos realizados para evaluar el rendimiento del modelo y su comparación con otros métodos de segmentación. Se realiza un análisis cuantitativo y cualitativo de los modelos, se realiza un estudio de ablación entre el modelo línea de base, UNet, y el propuesto, UNetFFT. Se compara la estimación de área de lagunas seleccionadas del modelo UNetFFT y áreas de referencia.

En el Capítulo 5: Discusión de resultados, se analizan los resultados obtenidos, identificando limitaciones y posibles causas en las diferencias de los resultados de las diferentes metodologías y se establece cómo la aplicación de filtros en el dominio de la frecuencia aportan la detección de detalles finos.

La presente tesis es el resultado de la investigación desarrollada en el artículo científico *Exploratory Analysis Using Deep Learning for Water-Body Segmentation of Peru's High-Mountain Remote Sensing Images* (Perez-Torres et al., 2024), publicado en la revista *Sensors* en 2024, que pertenece al cuartil Q1 según Scimago. Este trabajo sirvió como base para profundizar en el análisis y desarrollo del modelo de segmentación de lagunas en la cordillera del Vilcanota, abordando las limitaciones y ampliando los hallazgos presentados en la publicación. Así también, los resultados de este proyecto fueron expuestos en la XI Semana de Investigación, Innovación y Emprendimiento organizado por el Vicerrectorado de Investigación de la Universidad Nacional de San Antonio Abad del Cusco.

Capítulo 1

Generalidades

1.1. Planteamiento del Problema

1.1.1. Problemática

El agua es un recurso vital que cumple un rol fundamental en las sociedades y ecosistemas. Los Andes peruanos, considerados como ecosistema de alta montaña, son una fuente importante de agua dulce tanto para las propias montañas como para las regiones aguas abajo, debido principalmente a la presencia de glaciares, capas de nieve, ríos, lagos y humedales (Motschmann, 2021). Los Andes peruanos contienen el 70 % de glaciares tropicales del mundo, sin embargo, debido al cambio climático, estos glaciares han tenido retrocesos considerables en la últimas décadas, produciéndose así cambios espaciales tanto en los cuerpos de hielo como los de agua. Estos cambios modifican las dinámicas ambientales de los ecosistemas, impactando a las poblaciones en diferentes ámbitos como la agricultura, el acceso al agua potable, la generación de electricidad, actividades turísticas y produciendo desastres naturales (Wood et al., 2021).

Los Andes peruanos están constituidos por 20 cordilleras distribuidas en 14 departamentos, cada una de ellas está cubierta por glaciares y nevados cuyas extensiones se han visto reducidas hasta en un 56 % en las últimas seis décadas. Estos cambios han generado variaciones en los cuerpos de agua en las cuencas cercanas a estas cordilleras. En específico, el departamento del Cusco está atravesado por 4 cordilleras, a saber, la de Vilcabamba, Urubamba, Vilcanota y La Raya, por lo cual es el segundo departamento con mayor área glaciar del Perú, al albergar el

32 % del total de glaciares peruanos. Debido a la relación directa entre glaciares y lagunas, el Cusco también posee una gran cantidad de lagunas de origen glaciar, contando con alrededor de 1300 lagunas inventariadas y distribuidas entre las diferentes cordilleras, que se corresponden con un equivalente al 15 % del área total de lagunas a nivel nacional (INAIGEM, 2023).

Las lagunas en el Cusco son de una importancia vital, ya que estas se aprovechan para la agricultura, regadíos, para mantener y alimentar a los ganados (uso pecuario), también se utilizan en otras actividades como la acuicultura; además, algunas lagunas son atractivos turísticos que son fuente de ingresos para los habitantes locales, todas ellas son actividades económicas de las cuales depende parte de la población. Por otro lado, el recurso hídrico de las lagunas se utiliza para la producción de energía hidroeléctrica, además, algunas de ellas son fuente de agua potable. Alrededor del 30 % de la población del Cusco se beneficia directa o indirectamente de las lagunas de origen glacial (INAIGEM, 2020b).

Recientemente, se han detectado situaciones críticas respecto a las lagunas, es el caso de la laguna Piuray, la cual suministra de agua potable a aproximadamente el 40 % de la población del Cusco. Esta laguna ha sufrido las consecuencias del estrés hídrico debido al fenómeno de El Niño y la extracción de agua para consumo humano (Corrales, 2023), por lo que su nivel de agua se ha reducido de manera preocupante. Por otro lado, el retroceso glaciar empeora las condiciones de agua disponible para la población aguas abajo e influye en la formación de nuevas lagunas glaciares, muchas de ellas peligrosas. Por ello, el monitoreo regular de las lagunas es crucial para asegurar la seguridad del recurso hídrico en la región y prevenir posibles desastres. En este contexto, las imágenes satelitales se destacan como un instrumento importante para tareas de seguimiento y monitoreo del recurso hídrico, ya que permiten obtener datos actualizados y precisos sobre la evolución de las lagunas a lo largo del tiempo, así como identificar posibles cambios y amenazas. Las imágenes satelitales tienen la capacidad de cubrir grandes áreas regularmente, permitiendo obtener datos de regiones que son inaccesibles o zonas remotas, lo que las convierte en una opción sumamente importante para evaluar las lagunas de alta montaña de forma regular y a través del tiempo, permitiendo tomar decisiones adecuadas en su gestión y cuidado.

En 1972 se ejecutó la primera misión que lanzó al espacio un satélite especializado en

la observación terrestre y el seguimiento del planeta de forma masiva, el Landsat 1. Desde entonces se han llevado a cabo diversas misiones con la misma finalidad, lo que generó una gran disponibilidad de imágenes de la Tierra con alta resolución espacial y una frecuencia temporal de captura rápida, permitiendo una comprensión de los procesos terrestres y cambios en la superficie del planeta. Para realizar estos estudios, las primeras aproximaciones en el uso de las imágenes satelitales empleaban técnicas básicas de procesamiento de imágenes como mejoramiento de contraste y color, mientras que era necesaria la interpretación visual de estos resultados por un experto, quien realizaba las clasificaciones de su interés manualmente. Con el tiempo, se desarrollaron nuevas técnicas de procesamiento en las cuales se combinaban diferentes bandas espectrales de las imágenes para resaltar ciertos cuerpos con determinadas características, desarrollándose lo que se conoce como índices espectrales, tales como el índice de vegetación de diferencia normalizada (NDVI), índice de agua de diferencia normalizada (NDWI, McFeeters, 1996), entre otros. El uso de estos índices mejoran la precisión de los análisis de los expertos, sin embargo, debido a la variabilidad de los datos recogidos en la imágenes, la evaluación y determinación de los valores adecuados para los índices, se hace de forma manual, lo que significa una gran carga de trabajo cuando se pretende analizar grandes regiones y durante diferentes años. En la actualidad, muchos estudios que requieren la segmentación y análisis de vegetación, ciudades, carreteras, cuerpos de agua, entre otros elementos, aún se realizan utilizando estos índices espectrales, debido principalmente a su efectividad para diferenciar estos cuerpos respecto de su fondo. Sin embargo, este proceso es lento y tedioso debido al trabajo manual que se requiere para analizar gran cantidad de datos.

El avance y desarrollo de la inteligencia artificial, específicamente del aprendizaje profundo (*deep learning*), ha permitido la creación de nuevas estrategias para la segmentación de imágenes satelitales. Estas técnicas están en constante desarrollo, buscando mejorar la precisión y automatización de los estudios que requieren un análisis detallado de grandes volúmenes de datos. Uno de los enfoques desarrollados es el de la segmentación de cuerpos de agua que comprende la diferenciación en una imagen satelital de mares, ríos, arroyos, embalses, humedales, lagos y lagunas. Parte de la atención de la comunidad científica ha pasado por el estudio de lagos y lagunas, ya que se consideran como centinelas sensibles y rápidos frente a los cambios

climáticos e hidrológicos en los ecosistemas a los que pertenecen, permitiendo comprender las dinámicas ambientales (Adrian et al., 2009).

A pesar del desarrollo de estas nuevas técnicas de segmentación de cuerpos de agua, aún existen desafíos significativos en la eficiencia de estos métodos, especialmente en entornos montañosos como los Andes. Problemas como la dificultad para diferenciar el agua de las sombras, la detección precisa de los bordes y la limitada transferibilidad de los modelos entrenados a otras zonas con diferentes características geográficas persisten. Esto se ha evidenciado en estudios nacionales donde se han utilizado métodos más convencionales, como los índices espectrales clásicos para el estudio de lagunas de origen glaciar (INAIGEM, 2023). Sin embargo, la geografía única de los Andes, con su compleja topografía y variabilidad atmosférica, requiere soluciones más avanzadas y adaptadas a estas condiciones particulares. En este contexto, existe la necesidad de desarrollar un algoritmo o modelo de aprendizaje profundo específicamente diseñado para la segmentación de lagunas en entornos de alta montaña de la región del Cusco, la cual es la segunda región del Perú con mayor cantidad de espejos de agua, en específico, en la cordillera del Vilcanota. Este algoritmo debe abordar los desafíos mencionados, aprovechando la capacidad del aprendizaje profundo para aprender y adaptarse a patrones complejos en las imágenes satelitales, proporcionando una forma eficiente de obtener resultados precisos y robustos en un área geográfica tan compleja como son los Andes.

1.1.2. Formulación del Problema

Problema General

La cordillera del Vilcanota, ubicada en la región Cusco, Perú, presenta una gran cantidad de lagunas glaciares de alta montaña, cuyo monitoreo y medición son fundamentales debido a su importancia ambiental y a los impactos del cambio climático. Las imágenes capturadas mediante los satélites presentan pobre precisión en la predicción y detección borrosa de bordes de las lagunas cuando son procesadas con redes neuronales convolucionales, afectando en la estimación precisa y automática de sus áreas.

1.2. Objetivos

1.2.1. Objetivo General

Implementar un método automatizado de segmentación y determinación precisa de áreas de lagunas en la cordillera del Vilcanota, región Cusco, Perú, utilizando modelos de aprendizaje profundo adaptados a las características del entorno de alta montaña.

1.2.2. Objetivos Específicos

1. Implementar y entrenar un modelo de aprendizaje profundo capaz de segmentar con precisión las lagunas, diferenciándolas de otros elementos del terreno en el entorno de alta montaña.

2. Comparar cuantitativa y cualitativamente el desempeño del modelo propuesto con métodos tradicionales y otros modelos de segmentación, utilizando métricas de evaluación adecuadas.

3. Evaluar la exactitud del modelo en la estimación del área de las lagunas mediante la comparación con mediciones de estudios previos realizados en la región.

1.3. Hipótesis

1.3.1. Hipótesis General

La implementación de un modelo de segmentación basado en aprendizaje profundo permite la identificación y estimación precisa de áreas de lagunas en la cordillera del Vilcanota, región Cusco, adaptándose a las características geográficas y climáticas particulares del entorno de alta montaña.

1.3.2. Hipótesis Específicas

1. El modelo de segmentación basado en aprendizaje profundo diferencia lagunas de otros elementos de su entorno, siendo robusto frente a las condiciones geográficas y climáticas del entorno de alta montaña, asegurando resultados precisos y consistentes en la estimación del área de lagunas.

2. El modelo de aprendizaje profundo desarrollado presenta un desempeño superior en términos de las métricas de evaluación en comparación con métodos tradicionales y otros modelos de segmentación en el contexto de la cordillera del Vilcanota.

3. El modelo de segmentación propuesto demuestra exactitud en la estimación del área de las lagunas en la cordillera del Vilcanota respecto a las mediciones obtenidas en estudios previos realizados por instituciones reconocidas en la región.

1.4. Justificación

1.4.1. Justificación Social

El monitoreo con alta frecuencia temporal de lagunas es sumamente importante, ya que estos elementos naturales tienen un impacto directo en la vida de la población de la región del Cusco, ya que más del 30 % de la población cusqueña se beneficia directa o indirectamente de este recurso (INAIGEM, 2020b). Por un lado, la disminución del nivel de agua en estas lagunas, principalmente causada por la extracción de agua y el cambio climático, puede llevar a crisis hídricas afectando la salud de la población. Mientras que la expansión y generación de nuevas lagunas puede provocar desastres por estallido de lagunas glaciares, huaicos y deslizamientos. El monitoreo de lagos y lagunas con alta frecuencia temporal requiere la implementación de sistemas que utilicen métodos eficientes y confiables que puedan realizar estas tareas de forma automática. Por lo tanto, la utilización de imágenes satelitales y aprendizaje profundo en el marco de un sistema de monitoreo automatizado puede contribuir significativamente a una gestión del recurso hídrico que sea sostenible y al bienestar de las comunidades locales.

1.4.2. Justificación Tecnológica

La implementación de un algoritmo basado en aprendizaje profundo enfocado en la segmentación de lagunas en entornos de alta montaña es un aporte tecnológico para la rama de la teledetección. La aplicación de estas técnicas puede permitir la automatización en el análisis de imágenes satelitales, proporcionando resultados más precisos en comparación con métodos tradicionales. Además, la utilización de algoritmos de aprendizaje profundo, gracias a su capacidad de generalización, puede mejorar la capacidad de adaptación del modelo a diferentes condiciones geográficas y ambientales, siendo en una herramienta adaptable y flexible para la monitorización de recursos hídricos en regiones montañosas.

1.4.3. Justificación Académica

Este estudio se constituye como un aporte en el conocimiento de áreas como la teledetección y la inteligencia artificial. Esto ya que la implementación de un modelo de aprendizaje profundo en la segmentación de lagunas en contextos de entornos de alta montaña se enfoca en resolver algunos problemas conocidos de las metodologías actuales. Por otro lado, la posible aplicabilidad práctica de este modelo puede facilitar la tarea de monitoreo de lagunas en la región del Cusco.

1.4.4. Justificación Ambiental

Las lagunas pertenecientes a entornos de alta montaña de la región del Cusco son recursos importantes para la población cusqueña así como para la biodiversidad que existe cerca de estos lugares. El cambio climático afecta a los diferentes ecosistemas en el mundo, uno de ellos son los de alta montaña, los cuales son especialmente sensibles al aumento de las variaciones de temperatura. La degradación de estos ecosistemas puede afectar negativamente a las sociedades que dependen de este recurso. Resolver la tarea de segmentación de lagunas en estos contextos habilitará la posibilidad detectar y mitigar los efectos del cambio climático, permitiendo también comprender las dinámicas ambientales en estas zonas.

1.4.5. Justificación Económica

Dado que una cantidad considerable de la población cusqueña depende de las lagunas para efectuar sus actividades económicas tales como la agricultura, ganadería, turismo, entre otras, es evidente que el deterioro de este recurso afecta negativamente a la población. Por lo que el desarrollo de un modelo de segmentación de lagunas puede prevenir pérdidas económicas pudiendo tomarse a tiempo decisiones que promuevan el uso sostenible de este recurso.

1.5. Alcances

Dado que el objetivo de la investigación es implementar un algoritmo de aprendizaje profundo para la segmentación de lagunas y compararlo con otros modelos y métodos, y que se realizará un análisis de las diferencias entre los comportamientos utilizando las métricas de evaluación consideradas según la literatura, entonces, la interpretación de los resultados en el contexto estudiado de entornos de alta montaña implica una investigación con alcance de tipo explicativo.

Los alcances específicos del desarrollo de esta tesis abarcan los siguientes conceptos:

- La investigación se centrará en la segmentación de lagunas de la cordillera del Vilcanota en la región del Cusco, excluyendo otros cuerpos de agua, debido a la importancia de las lagunas tanto para la población como para el mismo ecosistema.
- Se elaborará un conjunto de datos específico que abarque la cordillera del Vilcanota considerando condiciones climáticas similares para las imágenes compiladas, como baja nubosidad y mismas fechas anuales en la recopilación de datos, para garantizar una consistencia en los experimentos.
- El modelo implementado para la segmentación de lagunas estará dentro del marco del paradigma del aprendizaje profundo.
- El rendimiento del modelo será comparado con otros modelos de segmentación de imágenes así como con métodos tradicionales para la segmentación de lagunas. Esta comparación se realizará para evaluar el comportamiento del modelo frente a técnicas establecidas.

- La exactitud en la estimación del área de lagunas se realizará mediante la comparación de la predicción del modelo para lagunas específicas dentro de la zona de estudio (cordillera del Vilcanota) seleccionada en base a estudios previos (INAIGEM, 2020c).

1.6. Limitaciones

- Dado el trabajo manual para la generación del conjunto de datos específico, la anotación de los datos puede estar sujeta a errores.
- Las condiciones climáticas de la zona pueden afectar los resultados, por lo que la investigación se limitará a la cordillera del Vilcanota en condiciones climáticas específicas como baja presencia de nubes en la escena (menor al 10 %), así como libre de neblina y mismo rango de fechas anuales de adquisición.
- Los resultados obtenidos por el modelo pueden no ser generalizables para otras zonas diferente al lugar de estudio, debido a diferencias geográficas y ambientales.
- A pesar de que las técnicas de aprendizaje profundo pueden ofrecer buenos resultados, también pueden presentar limitaciones en la segmentación de lagunas en ciertos contextos con características complejas.
- Los modelos de aprendizaje profundo suelen requerir recursos computacionales significativos, por lo que los experimentos que se realizarán y los modelos de comparación serán escogidos en función de los recursos computacionales disponibles. Para la investigación se cuenta con un equipo con una tarjeta gráfica de 5GB de memoria disponible.
- Para evaluar la exactitud de la estimación de área del modelo, las lagunas seleccionadas en la cordillera del Vilcanota estarán basadas en el estudio realizado por (INAIGEM, 2020c), considerando la disponibilidad y correspondencia temporal de los datos. La selección de estas lagunas para esta evaluación se determinará en función de la concordancia de las fechas de estudio entre los datos disponibles y los proporcionados por la institución de referencia.

1.7. Metodología

1.7.1. Tipo de Investigación

La investigación cuantitativa se caracteriza por ser secuencial y probatoria. Se siguen una serie de pasos con un orden riguroso, aunque pudiéndose redefinir alguna fase. El objetivo de la investigación es probar la hipótesis, para ello se planifica un diseño de investigación. En un contexto delimitado y específico, se miden las variables de interés para luego analizarlas mediante métodos estadísticos. Finalmente, se extraen conclusiones respecto a la hipótesis (Hernández S., 2014). En el marco de la investigación planteada, esta cumple con las características descritas, por lo que se concluye que la investigación es de tipo cuantitativa.

1.7.2. Población y Muestra

Población

Para la presente investigación la población está delimitada a las lagunas que pertenecen al entorno de alta montaña de la cordillera del Vilcanota en la región del Cusco, Perú, con alrededor de 580 lagunas.

Muestra

Para efectuar la evaluación respecto a la exactitud del modelo para la estimación del área de las lagunas, basados en el estudio previo de (INAIGEM, 2020c) que identifica el área de algunas lagunas con peligro latente en la cordillera del Vilcanota, se seleccionan las lagunas con denominaciones A-10, A-40, AN-10, AN-20, H-10, H-20, H-30, Y-10 y CH-10 en los años 2019 y 2020. El sustento de su elección y estudio se desarrolla en la Sección 4.6.

1.7.3. Variables e Indicadores

Variable Independiente

Área estimada de las lagunas en la cordillera del Vilcanota

- Dimensiones

    1. Área de lagunas en píxeles: La cantidad de píxeles clasificados como parte de las lagunas en las imágenes procesadas por el modelo.
    2. Área de lagunas en metros cuadrados: La superficie real estimada de las lagunas en unidades métricas estándar (metros cuadrados), derivada de la conversión de píxeles a área física.

Variables Dependientes

Precisión en la segmentación de lagunas a nivel de píxeles

- Dimensiones

    1. Porcentaje de píxeles correctamente clasificados como lagunas.

- Indicadores

    1. Métricas de evaluación en tareas de segmentación: MIoU (*Mean Intersection over Union*), *Pixel Accuracy* y *F1-Score*.

Exactitud del área estimada de la laguna seleccionada

- Dimensión

    1. Error absoluto y relativo entre el área estimada por el modelo y el área de referencia.

- Indicador

    1. Error absoluto calculado entre el área estimada y la referencia.
    2. Error relativo entre el área estimada y la referencia.
    3. Análisis estadístico no paramétrico mediante prueba de rangos con signos de Wilcoxon.

Variable Controlada

Conjunto de datos

- Dimensiones

    1. Características del conjunto de datos.

- Indicadores

    1. Resolución de las imágenes satelitales.
    2. Condiciones ambientales de las imágenes.
    3. Tamaño del conjunto de datos.
    4. Técnicas de preprocesamiento de datos.

1.7.4. Definición Operacional

Área Estimada de las Lagunas en la Cordillera del Vilcanota

- Es la variable independiente que representa el área estimada de lagunas en la cordillera del Vilcanota, obtenida a través del proceso de segmentación utilizando el modelo propuesto basado en aprendizaje profundo, esta se puede expresar en términos de cantidad de píxeles clasificados como lagunas, así como en términos de unidades métricas, tras una conversión o escalamiento de los píxeles.

Precisión en la Segmentación de Lagunas a Nivel de Píxeles

- Variable dependiente que evalúa el comportamiento de las predicciones, numéricamente se calcula utilizando las métricas estándar para la evaluación de tareas de segmentación *MIoU*, *Pixel Accuracy* y *F1-Score*, que se basan en los resultados de clasificación a nivel de píxeles.

Exactitud del Área Estimada de las Lagunas Seleccionadas

- Variable dependiente que representa la exactitud con la que el modelo de segmentación propuesto estima el área de una laguna específica en la cordillera del Vilcanota, comparada con el área de referencia proporcionada por una institución de confianza. Los resultados de exactitud se someten a la prueba de Wilcoxon para muestras pareadas, una prueba estadística no paramétrica que evalúa si hay una diferencia en la tendencia central de los datos evaluados.

Conjunto de Datos

- Características del conjunto de datos: Las imágenes para realizar el entrenamiento y validación provienen del ecosistema de alta montaña de la cordillera del Vilcanota, con baja o nulas condiciones de nubosidad o neblina. Las mismas técnicas de preprocesamiento se aplican para todos los modelos, permitiendo una consistencia en los experimentos.

1.7.5. Técnicas e Instrumentos

Técnicas

- Procesamiento de imágenes satelitales.
- Modelado y entrenamiento del modelo de aprendizaje profundo.
- Comparación con otras técnicas y modelos mediante métricas.
- Evaluación de exactitud del modelo y comparación estadística.

Instrumentos

- Entorno de desarrollo: Python.
- Biblioteca de aprendizaje profundo: Tensorflow.
- Hardware: estación de trabajo con GPU de 5GB de memoria.
- Imágenes satelitales obtenidas de fuentes abiertas.

1.7.6. Flujo de Trabajo

El núcleo de la presente investigación es la implementación de un modelo basado en aprendizaje profundo para la segmentación de lagunas en la cordillera del Vilcanota, la cual presenta un entorno altamente complejo y diverso. Parte fundamental en el desarrollo de un modelo de aprendizaje es la etapa de preprocesamiento de la información, ya que si los datos están correctamente formateados, transformados y organizados, el modelo puede aprender mejor las características intrínsecas de la información. En la Figura 1.1 se muestra el flujo de trabajo a seguir para realizar esta tarea de preprocesamiento. Para el proceso de implementación, entrenamiento y evaluación del modelo, se siguen las prácticas comunes para estas tareas en el campo de aprendizaje profundo. La Figura 1.2 representa el flujo de trabajo necesario para la implementación del modelo propuesto para segmentación de lagunas, el cual contempla un proceso iterativo para obtener resultados competitivos.

En la Figura 1.1, se observan los pasos requeridos para obtener el conjunto de datos específico para la zona de estudio. Como elemento de entrada se considera a las escenas satelitales multiespectrales provenientes del satélite Landsat-8, cuya justificación en su elección se muestra en la Sección 3.1.1. Se siguen otros pasos de procesamiento estándares para el manejo de imágenes satelitales como el apilamiento y enderezamiento de escenas. Basados en las escenas seleccionadas y procesadas, se crean los parches que son trozos más pequeños de la imagen entera para facilitar el procesamiento mediante los modelos de aprendizaje. Se generan las máscaras de lagunas asociadas a cada parche generado. Los valores de los parches se encuentran en número digital (*digital number*) de 16 bits de profundidad, estos se reescalan a valores de reflectancia superficial (SR), se manejan los datos faltantes y se normaliza cada banda de los parches. El conjunto de parches y máscaras forman el conjunto de datos para el entrenamiento, validación y evaluación del modelo propuesto y los de comparación.

En la Figura 1.2 se muestran los pasos requeridos para la implementación del modelo propuesto. Ya que la propuesta es una modificación al modelo UNet que mejore sus resultados en la segmentación de lagunas, se tiene un proceso iterativo en el que se añade el módulo propuesto y se realizan diversas modificaciones en sus parámetros para obtener mejoras en el

Figura 1.1

Diagrama de flujo para preprocesamiento de los datos.

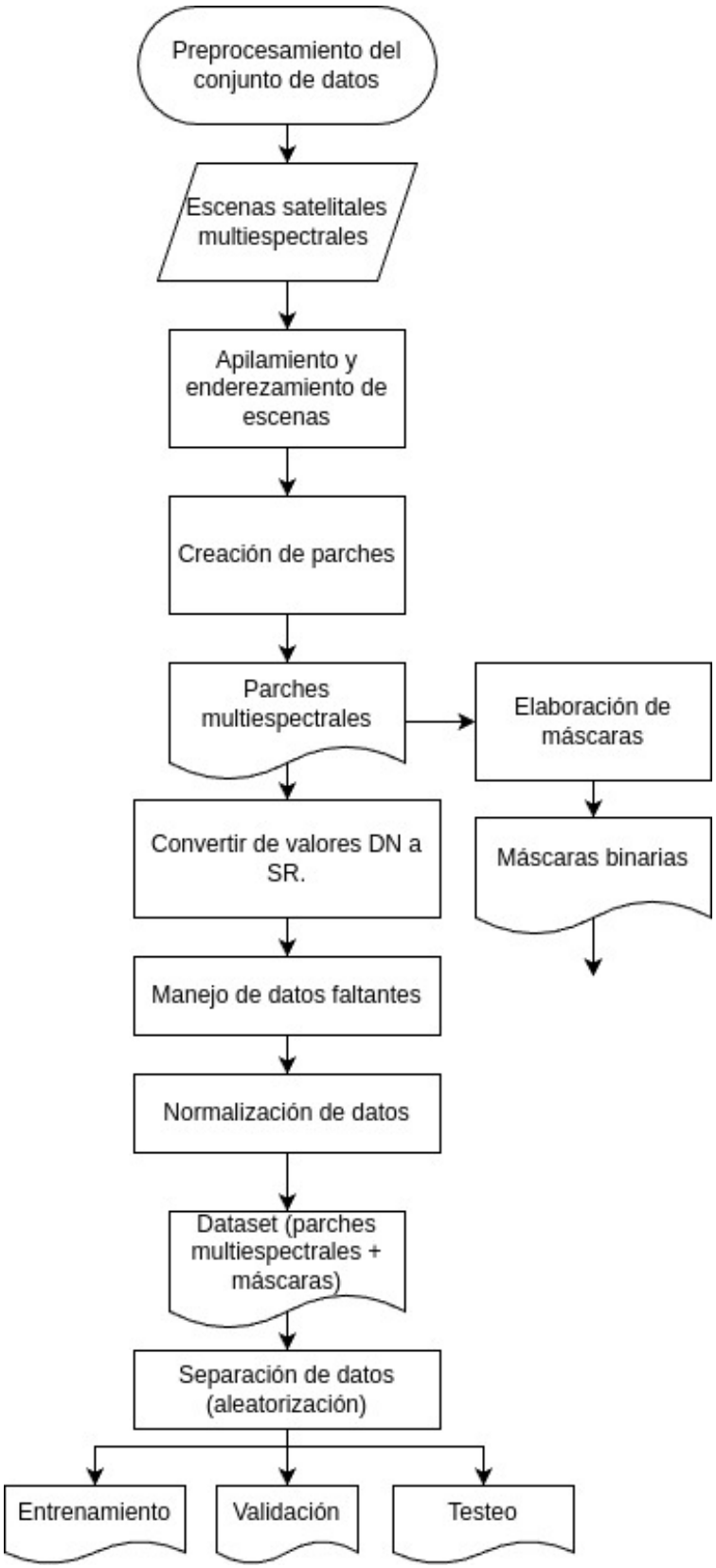
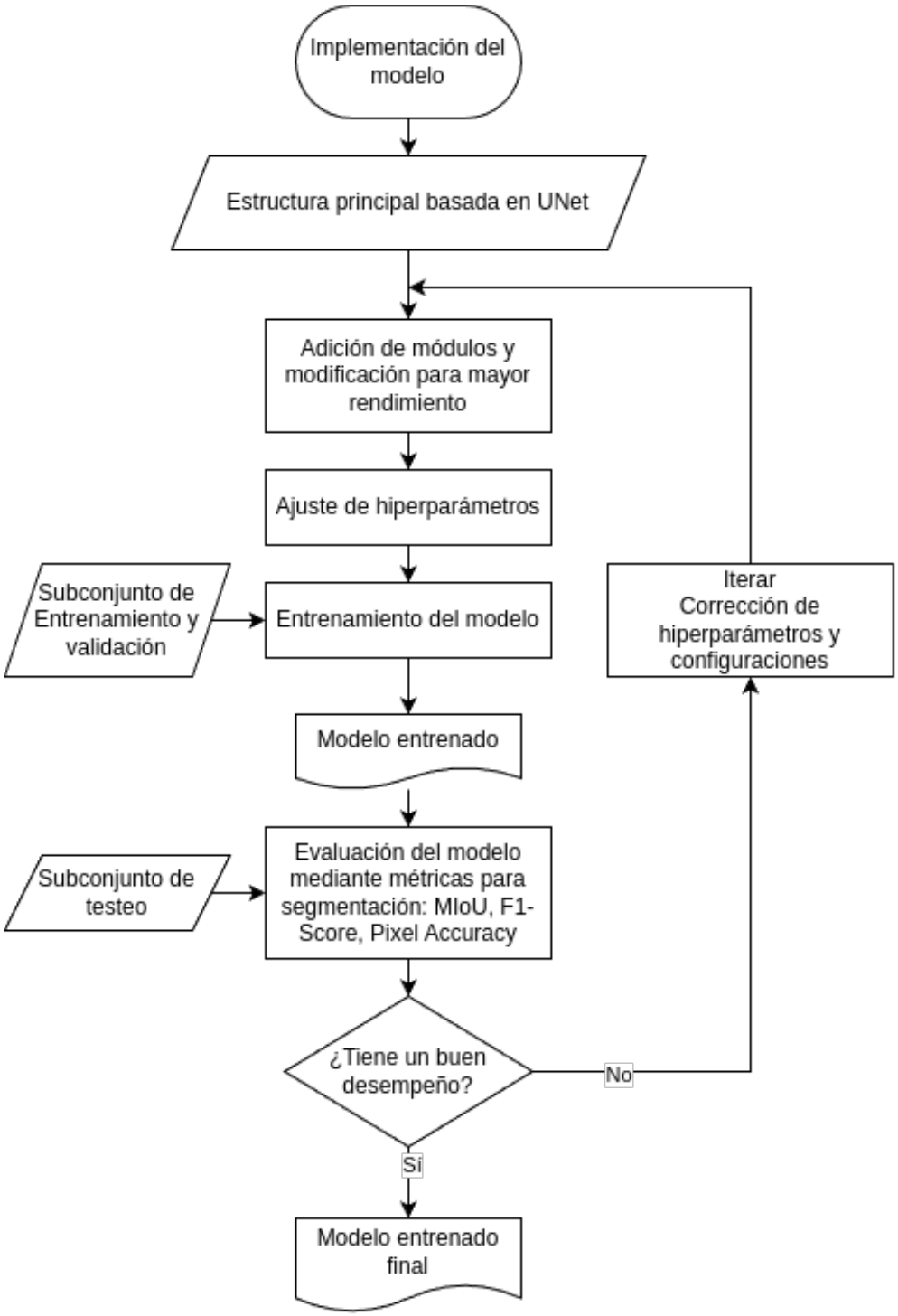


Figura 1.2

Diagrama de flujo para implementación del modelo de Aprendizaje.



rendimiento del modelo línea de base. Se evalúa su rendimiento basados en métricas estándar para la segmentación de imágenes según la literatura. Cuando se obtiene un modelo que consistentemente obtiene mejores resultados en la tarea de segmentar lagunas, se termina el flujo de implementación de la propuesta.

Con estos resultados, se procede a realizar la comparación del modelo propuesto con otras metodologías de comparación, tales como otros modelos especializados en la tarea de segmentación (con su respectivo entrenamiento) así como métodos tradicionales para la segmentación de lagunas como la utilización de índices espectrales (NDWI). Se evalúa la significancia estadística de la mejora hallada en el modelo propuesto mediante la prueba de rangos con signos de Wilcoxon (recomendado según la literatura, ver Sección 2.11.1). Finalmente, se evalúa la exactitud en la estimación de área del mejor entrenamiento del modelo propuesto comparado con mediciones de área realizadas por una institución de confianza, en este caso, INAIGEM.

Capítulo 2

Marco Teórico

2.1. Antecedentes

A partir del lanzamiento de los primeros satélites destinados a la recolección de información óptica terrestre en 1972 con el satélite Landsat 1, la comunidad científica ha realizado esfuerzos para estudiar las dinámicas de los diferentes recursos disponibles en la superficie de la Tierra. Uno de los focos de atención son las masas de agua, tales como mares, ríos, lagunas y humedales. Aunque los primeros enfoques para evaluar este recurso natural consistían en técnicas clásicas de procesamiento de imágenes como aplicación de filtros y mejoramiento de contraste para detectar patrones y zonas homogéneas; enfoques más recientes han incorporado técnicas más avanzadas como el aprendizaje automático. Estas técnicas de vanguardia incluyen la utilización de estrategias basadas en aprendizaje profundo como las redes neuronales convolucionales, mecanismos *transformers*, así como algoritmos de atención. Así también, se aplican técnicas de procesamiento como el análisis espectral de las imágenes.

Para enfrentar la tarea de segmentación y detección de lagunas, los métodos basados en índices espectrales todavía son ampliamente utilizados debido a su capacidad para analizar la información contenida en imágenes satelitales de forma precisa. Sin embargo, presentan dificultades relacionadas a la gran carga de trabajo manual que requieren para caracterizar cada imagen satelital y diferenciar adecuadamente los cuerpos de agua, esto debido a la variabilidad del entorno el cual suele ser heterogéneo y sujeto a cambios drásticos en el tiempo.

En el contexto académico peruano se destaca la tesis titulada **Identificación y registro catastral de cuerpos de agua mediante técnicas de procesamiento digital de imágenes en Landsat-5** presentada por (Laura y Alvarez, 2013), en la cual se desarrolla una aplicación que permite el registro catastral de cuerpos de agua de forma automatizada, para ello, se emplean técnicas de procesamiento de imágenes tales como las operaciones morfológicas, técnicas de binarización y representaciones en falso color. La metodología empleada consiste en efectuar las correcciones radiométricas y atmosféricas a los datos crudos obtenidos del satélite Landsat-5, para luego obtener los índices espectrales de las escenas de interés, a saber: índice de vegetación de diferencia normalizada (NDVI), índice de agua de diferencia normalizada (NDWI) e índice de agua de diferencia normalizada mejorado (MNDWI). Se obtienen los límites del área de la laguna segmentada mediante la aplicación de operaciones morfológicas.

En el ámbito nacional se destaca la reciente **Memoria Descriptiva del Inventario Nacional de Glaciares y Lagunas de Origen Glaciar 2023** publicada por el Instituto Nacional de Investigación en Glaciares y Ecosistemas de Montaña (INAIGEM, 2023). En dicho informe, entre otras consideraciones, se alcanza información sobre las lagunas de todas las cordilleras del Perú y que son de origen glaciar. Como metodología de trabajo, este estudio utilizó las imágenes satelitales brindadas por la misión Sentinel-2, la cual posee una cámara multiespectral de 13 bandas, una amplia cobertura que abarca 290 km de ancho y una gran resolución espacial de 10 m por píxel. Como software principal para el estudio se consideró tanto a Python como JavaScript, así como la herramienta de Google Earth Engine. Respecto al procesamiento y detección de las coberturas de nieve y agua se utilizó el índice de nieve de diferencia normalizada (NDSI) y el índice de agua de diferencia normalizada (NDWI). La aplicación de estos índices consideró la selección de umbrales que diferencian la presencia o no de los elementos indicados. Dado que la aplicación directa de los índices no son siempre confiables, en el estudio se realizó una exhaustiva verificación, ajuste y corrección manual de las lagunas y glaciares, utilizando para ello imágenes de alta resolución.

En un contexto internacional, en el artículo **Influence of natural factors and land use change on changes in the main lake area in China over the past 30 years** publicado por (Y. Wang et al., 2023) se realiza un estudio sobre 38 lagos ubicados en China. Para ello, emplea las

imágenes provenientes de las misiones Landsat en el periodo comprendido de 1989 a 2019, las imágenes fueron obtenidas en el intervalo de meses de noviembre a abril por ser los meses donde no existe nubes en esa zona. La metodología empleada para realizar el análisis del cambio en las áreas de los lagos estudiados consiste en la aplicación del índice NDWI junto con método de selección de umbral automático y adaptativo, Otsu. Se identificó que la aplicación de umbrales uniformes en el algoritmo Otsu en la extracción de cuerpos de agua provoca ciertos errores: si el umbral es bajo, los cuerpos de agua se extraerán con ciertos tipos de tierra, mientras que si el umbral es demasiado alto, se perderán cuerpos de agua.

Al respecto de la segmentación de lagos y lagunas con enfoques basados en aprendizaje profundo, existe amplia literatura que aborda diferentes enfoques para la tarea. Las arquitecturas para procesamiento de imágenes, en específico para la segmentación semántica, bajo el paradigma del aprendizaje profundo han adquirido importancia desde la propuesta de las redes neuronales convolucionales (CNN), surgiendo redes totalmente convolucionales (Fully Convolutional Networks, FCN), redes de tipo UNet basadas en capas convolucionales así como otras variaciones que introducen módulos de atención y de aumento de campo perceptivo.

Variantes de estos modelos se han desarrollado para la tarea de segmentación de cuerpos de agua, lagos y lagunas. El artículo **Water Body Extraction from Very High Spatial Resolution Remote Sensing Data Based on Fully Convolutional Networks** presentado por (L. Li et al., 2019), presenta una modificación de una FCN para la extracción de cuerpos de agua en una zona metropolitana de Beijing. En el estudio se hace un análisis del efecto de diferentes factores en el rendimiento del modelo, estos factores abarcan ciertas configuraciones en las imágenes de entrada, el aumento de datos y la transferencia de aprendizaje. Las imágenes para el estudio son obtenidas del satélite Gaofen-2, el cual ofrece imágenes en 4 bandas: azul, verde, rojo e infrarrojo cercano. Para el contexto de área urbana estudiado, la red presenta un comportamiento robusto, obteniendo mejores resultados que el método basado en NDWI y superando también a otros métodos clásicos de *machine learning* (ML). En el artículo se plantean limitaciones ligadas a la transferibilidad de esta red a otros contextos, por ejemplo, zonas montañosas, dadas las diferencias existentes entre las sombras de montañas y edificios.

El artículo **Water Surface Mapping from Sentinel-1 Imagery Based on Attention-**

UNet3+: A Case Study of Poyang Lake Region presentando por (Jiang et al., 2022), implementa una modificación de la arquitectura UNet, adicionando módulos de ateción espacial. Su estructura UNet, codificador-decodificador, permite que el modelo aprenda características jerárquicas de las imágenes. Para el estudio se utilizan datos provenientes de Sentinel-1, que son generados por radar de apertura sintética (SAR) y se analiza el rendimiento del modelo específicamente la zona del lago Poyang, en China. Los resultados del modelo superan a otras metodologías como la segmentación por umbralización y otras arquitecturas clásicas en la tarea de segmentación como UNet, DeeplabV3 y SegNet.

Se han desarrollado también enfoques que combinan diferentes arquitecturas, como **An applicable and automatic method for earth surface water mapping based on multispectral images** presentada por (Luo et al., 2021), implementan un modelo híbrido de tipo codificador-decodificador, utiliza como columna vertebral la arquitectura de MobilenetV2 para la parte de codificador, mientras que utiliza en el decodificador módulos de *atrous spatial pyramid pooling* (ASPP), pertenecientes a la arquitectura de DeepLabV3+, para obtener información espacial a múltiples escalas. El estudio utiliza los datos brindados por el satélite Sentinel-2, el cuan brinda imágenes de hasta 13 bandas espectrales, de las cuales, el estudio solo utiliza seis. Sobre el proceso de entrenamiento, se construyó una nueva base de conocimiento con aguas superficiales de todo el mundo. Se realizó comparaciones con otros modelos como DeepWaterMap, DeeplabV3+, además de algoritmos como *Support Vector Machine* (SVM) y metodologías basadas en MNDWI, obteniendo resultados superiores a cada uno de ellos. El análisis se separó en entornos urbanos y montañosos.

Recientemente, con la introducción de nuevos paradigmas en el aprendizaje profundo como los *transformers*, se han postulado alternativas que combinan redes CNN con los mecanismos *transformer*. El artículo **Water–land segmentation via structure-aware CNN–transformer network on large-scale SAR data** propuesto por (Y. Zhou et al., 2022) implementa un red basada en CNN con un transformador para la segmentación de agua-tierra en imágenes de SAR. Se utiliza un *transformer* para extraer información global de la imagen, mientras que la estructura de la red CNN en forma de U se utiliza para la segmentación precisa. Adicionalmente, se implementa un módulo de postproocesamiento basado en campos aleatorios completamente

conectados (CRF) para refinar la extracción de los bordes.

2.2. Lagunas: Importancia, Tipos y Cambios en el Tiempo

2.2.1. Importancia y Formación de Lagunas

Se define como lago a un cuerpo de agua abierto, es decir, que está expuesto a la atmósfera y puede recibir aporte de arroyos, ríos, precipitaciones o deshielo, y que no está conectado con el océano. Esta definición abarca tanto a los cuerpos de agua amplios o lagos, como a cuerpos de agua más pequeños o lagunas, así como a humedales y estanques profundos, ya que no existe una característica clara que las diferencie (Dodds, 2002). Por lo que en este estudio, se considera el término laguna como más adecuado para el contexto de estudio aunque los términos pueden ser intercambiables. Las lagunas tienen un papel crucial en el desarrollo de los ecosistemas a su alrededor, en específico, a los seres humanos nos brindan alimentos mediante la pesca, recreación, agua potable y paisajes admirables. Debido a su importancia vital en tantos aspectos, en el último siglo estos ecosistemas han sido estudiados intensamente.

En el mundo existen más lagos pequeños ($<1 km^2$) que grandes, siendo la mayoría de origen glacial (Seekell et al., 2021). Los lagos se originan debido a diferentes procesos geológicos, en términos generales estos son: movimientos tectónicos de la Tierra, actividad glacial y volcánica, erosión de diversas fuentes. Los lagos más extensos, profundos y antiguos del mundo se formaron debido a los movimientos tectónicos, este tipo de lagos, aunque pocos, cubren áreas y tienen volúmenes mayores a las lagunas pequeñas. La actividad glacial, principalmente en regiones templadas, ha formado y sigue formando muchos más lagos que cualquier otro proceso geológico, ya que muchos procesos asociados con la actividad glacial llevan a la formación de lagos. Finalmente, la actividad volcánica tiende a formar lagos, debido a las explosiones que dejan detrás cráteres que se pueden llenar de agua (Dodds, 2002).

2.2.2. Variables que Determinan el Estado de las Lagunas

Los lagos y lagunas tienen estructuras complejas y pueden ser caracterizados por diferentes variables que abarcan conceptos físicos, químicos y biológicos. Considerando estas condiciones

(Johnes et al., 1994) elaboraron una tabla con las características y variables principales que caracterizan a un cuerpo de agua dulce. Esta distribución se muestra en la Tabla 2.1.

Tabla 2.1
Variables que caracterizan cuerpos de agua dulce.

Morfometría, hidrología y geografía	**Química principal de iones**	**Disponibilidad de nutrientes**	**Azar, accidental y biológica**
Altitud	Geología local	Geología local	Grado de aislamiento
Latitud	Proximidad al mar	Vegetación de la cuenca	Inundaciones y sequías pasadas
Área, profundidad y volumen	Clima	Uso de tierra	Clima fluctuante
Precipitación, evaporación y tiempo de retención	Sedimentos	Sedimentos	Perturbación por eventos naturales y actividad humana

Nota: Tabla modificada de (Johnes et al., 1994).

El tamaño, profundidad y área del lago, su forma en planos verticales y horizontales, así como la presencia de sedimentos determinan algunas características principales de los lagos.

2.2.3. Cambios a lo Largo del Tiempo

Los lagos no son estructuras permanentes en el tiempo, estos son afectados por diferentes fenómenos y dinámicas ambientales propias de su ubicación geográfica, tales como la presencia de vegetación en sus márgenes, fuentes de agua como arroyos o deshielo de glaciares cercanos, precipitaciones y actividades humanas. Los lagos son especialmente sensibles a los cambios que se producen en las cuencas a las que pertenecen. Estas continuamente están en cambio, incluso con ausencia de actividad humana, aunque estas modificaciones se han visto intensificadas por el cambio climático. Los lagos siempre están cambiando en el tiempo, dependiendo de características naturales de sus cuencas, variando a diferentes velocidades pudiendo cambiar la tendencia del cambio en cualquier momento. Por estos motivos, a pesar que las condiciones actuales de los lagos son importantes, estos no pueden ser usados de forma aislada para elaborar conclusiones sin tomar en cuenta su evolución histórica (Johnes et al., 1994).

2.3. Cordillera del Vilcanota

Ubicación Geográfica del Área de Estudio

La cordillera del Vilcanota, ubicada en el sur de Perú, se extiende en una franja de 134 km de longitud y ocupa un área aproximada de 7,521 km². Sus coordenadas geográficas van desde los 14°33’08.86"hasta los 13°07’23.82"de latitud sur, y de 71°45’11.64.ª 70°28’14.91"de longitud oeste. Políticamente, su extensión abarca parte del departamento de Cusco, en las provincias de Paucartambo, Quispicanchi y Canchis (INAIGEM, 2018). En la Figura 2.1, se observa la ubicación geográfica de la cordillera del Vilcanota.

Como se observa en la Figura 2.1, la cordillera del Vilcanota abarca grandes extensiones de nevados y lagunas distribuidas la mayoría en el departamento del Cusco.

Importancia para el Estudio

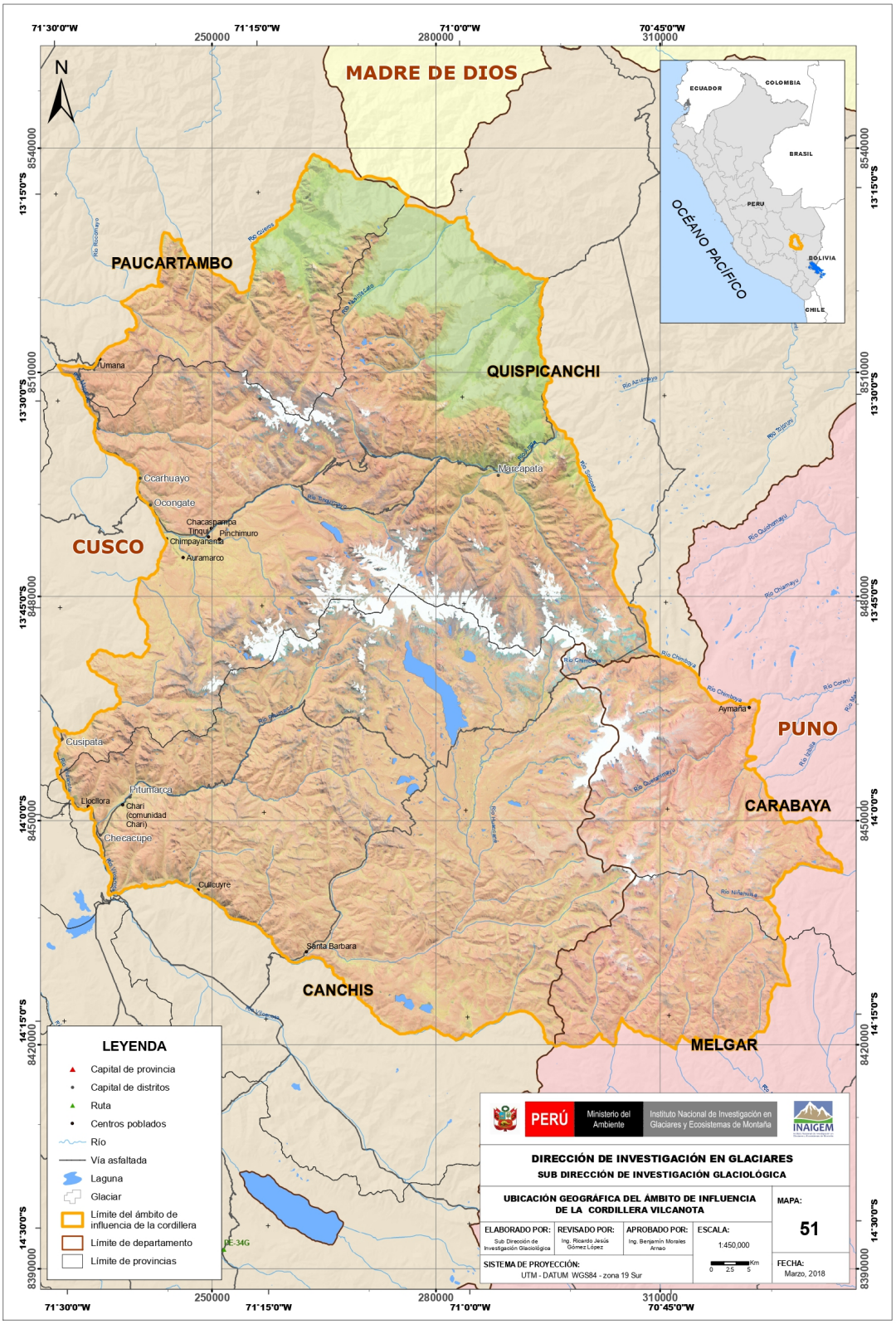
(INAIGEM, 2023) indica que la cordillera del Vilcanota es una de las que contiene mayor cobertura glacial. Se ha evidenciado que las cordilleras con nevados más amplios son las que tienen mayor tendencia a su reducción. Estas variaciones en la cobertura glacial hacen que los comportamientos de las lagunas de esta cordillera sufran modificaciones y cambios en el tiempo, por ejemplo, INAIGEM ha detectado que la cordillera del Vilcanota es la segunda con mayor número de lagunas nuevas para el 2023 (ver Figura 2.2.) Por lo que estudiar la cordillera del Vilcanota es fundamental para comprender las dinámicas, cambios y evolución de los cuerpos de agua, que son importantes para la gestión de recursos hídricos en la región.

Tipos de Lagunas en la Cordillera del Vilcanota

En general, los lagos / lagunas en las regiones glaciales a menudo tienen una variedad de componentes físicos y biológicos que influencian en sus características, como el color, que es una variable importante para su detección remota. El Instituto Nacional de Investigación en Glaciares y Ecosistemas de Montaña (INAIGEM, 2023) considera que la cordillera del Vilcanota contiene principalmente lagunas de origen glacial, las cuales se clasifican en tres: Lagunas periglaciares, proglaciares y supraglaciares.

Figura 2.1

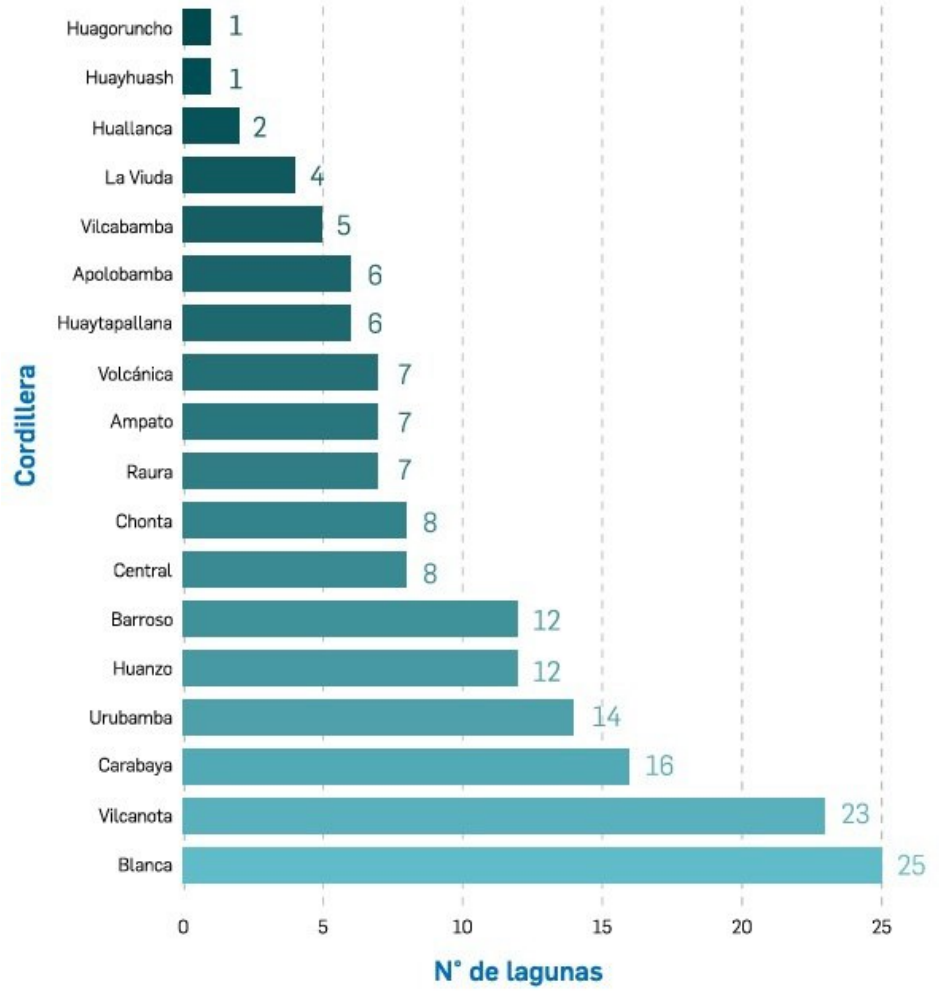
Ubicación geográfica de la cordillera del Vilcanota.



Nota: Extraído de (INAIGEM, 2018).

Figura 2.2

Nuevas lagunas por cordillera.



Nota: Extraído de (INAIGEM, 2023).

- Lagunas periglaciares: se consideran aquellas que en la actualidad se ubican donde antes se encontraban glaciares. No tienen contacto actualmente con el glaciar.
- Lagunas proglaciares: los cuerpos de agua que se encuentran al borde del glaciar. Tiene contacto directo con el cuerpo de hielo.
- Lagunas supraglaciares: las que se forman debido al agua de lluvia o deshielo que se desarrollan dentro de la superficie glaciar.

2.4. Imágenes Satelitales y Tecnología Satelital

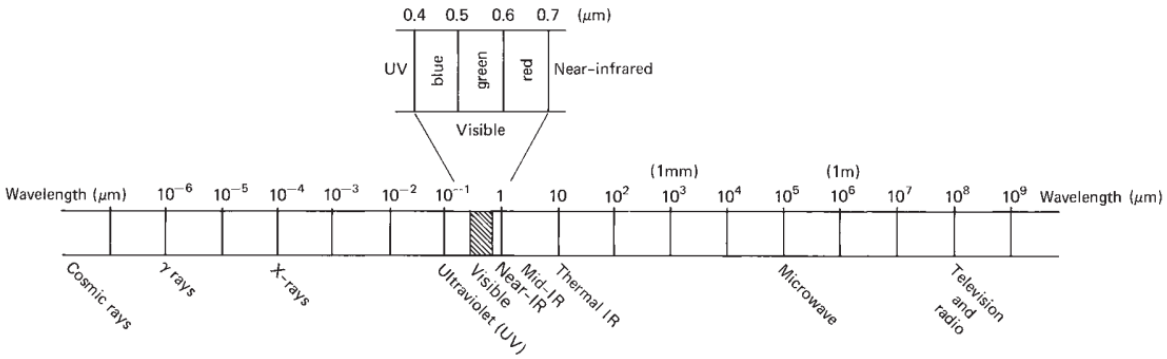
Las imágenes satelitales son un recurso fundamental para el estudio a gran escala de muchos fenómenos terrestres. Estas imágenes son capturadas por diferentes sensores que son sensibles a frecuencias del espectro electromagnético, pudiendo adquirir información incluso fuera del rango visible.

2.4.1. Radiación Electromagnética

Se define espectro electromagnético al conjunto continuo de ondas electromagnéticas que son emitidas o absorbidas por cualquier elemento. En el contexto de las imágenes satelitales, el Sol se constituye la principal fuente de radiación electromagnética. Una de las formas de graficar el espectro electromagnético es mediante un eje horizontal que representa la longitud de onda (*wavelength*), muchas veces expresada en μm.

Los ojos de los humanos son sensibles a una porción del espectro electromagnético, por lo que esta parte es conocida como espectro visible, que abarca las longitudes de onda desde 0.4 μm hasta 0.7 μm. Las radiaciones más cercanas a este espectro son la ultravioleta, cuya longitud de onda es menor que el espectro del color azul; y el infrarrojo, cuya longitud de onda es más amplia que la del color rojo. La distribución del espectro electromagnético se observa en la Figura 2.3.

Figura 2.3
Espectro electromagnético (μm).



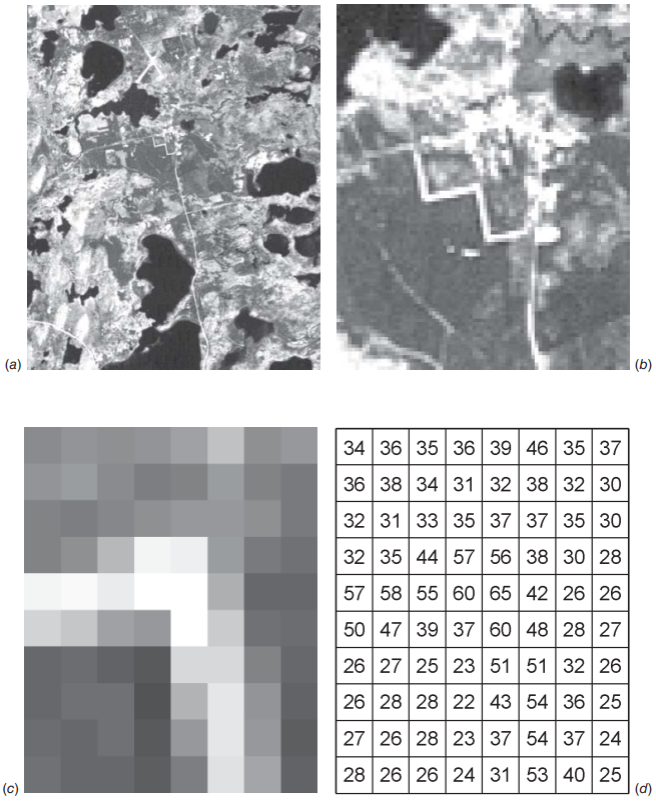
Nota: Extraído de (Lillesand et al., 2015).

2.4.2. Adquisición de Imágenes Digitales

Antes de la llegada de los sensores electrónicos, se utilizaba el film fotográfico, cuyo funcionamiento se basaba en el aprovechamiento de reacciones químicas sensibles a determinadas ondas electromagnéticas, proceso que resultaba ser de tipo analógico. Sin embargo, con el desarrollo de los sensores electrónicos, la forma adquirir las imágenes cambió. Estos sensores se diseñaron para ser sensibles a determinadas variaciones en la energía de una escena y generar una señal eléctrica correspondiente. Este proceso es principalmente de tipo digital (Lillesand et al., 2015).

En la Figura 2.4, se observa en (a) la captura de una escena de la Tierra mediante tecnología digital, en (b) y en (c) una visión acercada de una zona de la imagen y en (d) los valores digitales de la imagen (c).

Figura 2.4
Captura de una imagen digital satelital.



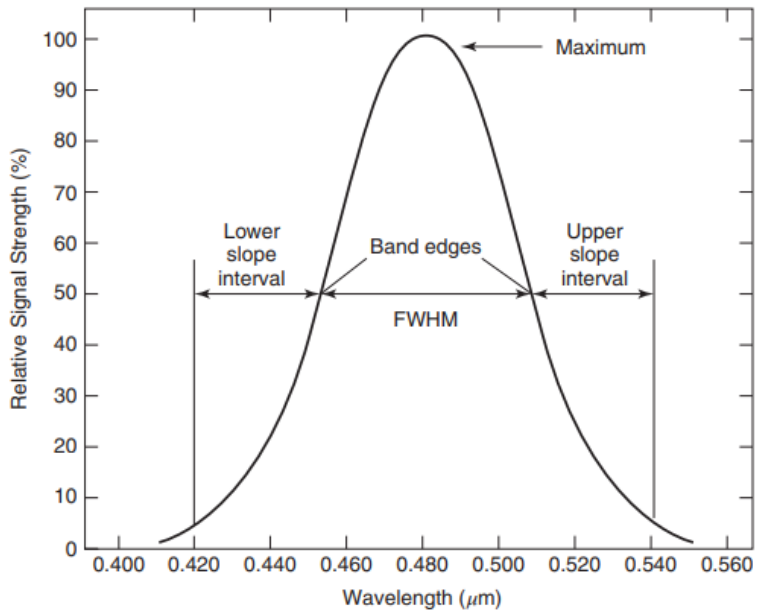
Nota: Extraído de (Lillesand et al., 2015).

2.4.3. Tecnología Satelital

Los satélites están compuestos por dos subsistemas principales: el *bus* y el *mission system*, siendo el primero la estructura básica que proporciona soporte, energía y control al satélite y el segundo los instrumentos y sensores específicos diseñados para cumplir con los objetivos de la misión del satélite. Algunos satélites están equipados con múltiples sensores para capturar escenas de la Tierra, los cuales, en general, son sensibles a determinadas bandas espectrales. Esta sensibilidad se expresa usando la terminología *full width, half maximum* (FWHM) (Lillesand et al., 2015).

En la Figura 2.5 se observa el diagrama FWHM para un sensor sensible para la luz azul (0.480 μm). El valor de FWHM indica cuán sensible es un sensor a un determinado rango de longitudes de onda. Cuanto menor sea el valor de FWHM, mejor será la sensibilidad a una banda muy estrecha y específica.

Figura 2.5
FWHM para espectro de luz azul.



Nota: Extraído de (Lillesand et al., 2015).

Los sensores utilizados en satélites son caracterizados mediante FWHM, ya que determina la capacidad de los instrumentos para capturar información en diferentes longitudes de onda. Existen diversos tipos de sensores en satélites que operan en múltiples bandas espectrales, cada uno optimizado para aplicaciones específicas.

2.4.4. Satélites para Adquisición de Imágenes

Existen diferentes satélites que se han desplegado desde 1972 con la finalidad de observar la Tierra, siendo los primeros los pertenecientes a las misiones Landsat. Luego, han sido desplegados otros satélites con características específicas para diferentes propósitos.

En la Tabla 2.2, se muestran diferentes satélites lanzados para la adquisición de imágenes superficiales de la Tierra, se detallan las bandas espectrales que recopilan, la resolución espacial (tamaño de píxel estimado) y la disponibilidad de estos recursos.

Tabla 2.2
Diferentes satélites y características de las imágenes que recopilan.

Satélite	**Bandas**	**Resolución por píxel**	**Disponi.**	**Acceso**
Landsat-1, 2, 3	B, G, R, NIR	80m	1972 - 1983	Libre
Landsat-4, 5	B, G, R, NIR, SWIR, TIRS	30m	1982 - 2013	Libre
Landsat 7	B, G, R, NIR, SWIR1, SWIR2, TIRS, PAN	30m	1999 - 2024	Libre
Landsat 8	B, G, R, NIR, SWIR1, SWIR2, TIRS, Cirrus, Aerosol	30m	2013 - p.	Libre
Landsat 9	B, G, R, NIR, SWIR1, SWIR2, TIRS, Cirrus, Aerosol	30m	2021 - p.	Libre
Sentinel-2	B2, B3, B4, B5, B6, B7, B8, B8A, B11, B12	10-20m	2015 - p.	Libre
MODIS	36 bandas (multiespectral)	250-1000m	2000 - p.	Libre
WorldView-2	B, G, R, NIR, Coastal	0.5-2m	2009 - p.	Restring.

Nota: B: azul, G: verde, R: rojo, NIR: infrarrojo cercano, SWIR: infrarrojo de onda corta, TIRS: infrarrojo térmico.

Existen múltiples satélites con diferentes características, tanto en las bandas que son capaces de capturar como en la resolución de las imágenes. Se observa también que Landsat es el conjunto de misiones más antiguo y con mayor cantidad de datos (desde 1972), aunque no todos sus sensores y bandas capturadas son compatibles en cada misión. Por otro lado, los satélites más recientes tienen mayores resoluciones. La mayoría de estas imágenes producidas por los satélites son de acceso gratuito a través de sus respectivas plataformas.

2.5. Segmentación de Imágenes

En el rubro de visión por computadora, uno de los problemas clásicos es la segmentación de imágenes. La segmentación de imágenes se considera un problema en el que se debe asignar a cada píxel de la imagen una determinada categoría inequívocamente (Terven et al., 2023), por lo que resulta en un problema de clasificación a nivel de píxeles. Existen diferentes métodos para realizar la segmentación de imágenes. La forma más sencilla y básica es la umbralización, la cual consiste en separar los píxeles en clases basados en la determinación de un valor umbral, normalmente esta separación se hace diferenciando la intensidad de los píxeles. Los métodos más actuales utilizan técnicas como el aprendizaje profundo que puede incluso diferenciar entre múltiples clases. La idea detrás de la segmentación es encontrar en la imagen partes o zonas con significado. El resultado de este proceso es la consecución una imagen en la cual las clases de interés ocupan toda la imagen, dando un significado a cada uno de los píxeles de la imagen.

La segmentación de imágenes se puede dividir en 3 diferentes categorías (Terven et al., 2023):

1. Segmentación semántica: consiste en segmentar una imagen en elementos incontables. Solo determina si un píxel pertenece a una determinada clase.

2. Segmentación de instancias: consiste en segmentar un elemento de una imagen y adicionarle a su máscara una etiqueta numérica que determina la cantidad de instancias que de tal elemento existen o se identifican en la imagen.

3. Segmentación panóptica: es una visión más completa del problema de segmentación en la cual se segmenta cada elemento de la imagen en una determinada clase y, además, identifica la cantidad de instancias de cada clase.

2.5.1. Aplicaciones de la Segmentación Semántica

La segmentación semántica es aplicada en diversos contextos y campos de la ciencia, tales como las imágenes médicas, la teledetección (*remote sensing*), así como el procesamiento de imágenes y video general.

En el campo de las imágenes médicas, existen múltiples aplicaciones que se han desarrollado para ayudar principalmente en el diagnóstico de determinadas enfermedades. Las imágenes médicas para procesar provienen principalmente de rayos X, imágenes de resonancias electromagnéticas, imágenes de muestras de biopsias, imágenes generadas por ultrasonido, entre otras. La segmentación de este tipo de imágenes con las técnicas clásicas como la umbralización (*thresholding*) y el agrupamiento (*clustering*) necesita de expertos que verifiquen los resultados (Patil y Deore, 2013), por ello, las técnicas de aprendizaje profundo han adquirido relevancia en el tiempo. Uno de los avances más importantes en el campo de las imágenes médicas fue el desarrollo del modelo de aprendizaje profundo UNet (Ronneberger et al., 2015) que propuso una red neuronal profunda basada en convoluciones que mejoró los resultados del estado del arte en la segmentación de imágenes médicas.

La teledetección es un campo que a partir de 1972 adquirió especial relevancia, debido a la disponibilidad de imágenes satelitales de gran resolución espacial. Además de estas imágenes, también se utilizan técnicas como los radares o tecnologías LiDAR. En sus inicios, estas técnicas consistían en la utilización de índices espectrales, los cuales operan con diferentes bandas espectrales generando nuevas imágenes que resaltan determinados cuerpos con ciertas características. Entre estos índices, los más comúnmente utilizados son el índice de vegetación de diferencia normalizada, NDVI (Kriegler, 1969); el índice de agua de diferencia normalizada, NDWI (McFeeters, 1996); el índice de nieve de diferencia normalizada, NDSI (Dozier y Painter, 2004), entre otros. A pesar de que estos métodos pueden llegar a ser muy precisos, mucho trabajo manual es requerido en la determinación de los umbrales para la segmentación de estas imágenes. Por ello, con el auge de los métodos basados en aprendizaje profundo, así como con la disponibilidad de gran cantidad de imágenes satelitales, se han desarrollado diversos modelos de segmentación semántica para la teledetección. Entre sus aplicaciones más comunes están la segmentación de agua (Yuan et al., 2021), segmentación de cultivos (H. Wang et al., 2022), segmentación de edificios (S. Zhou et al., 2023), entre otros. Sin embargo, a pesar de existir una cantidad ingente de información satelital, la disponibilidad de conjuntos de datos extensos y detallados aún es una limitación conocida.

En el procesamiento de imágenes y video en general se han desarrollado grandes avances,

debido principalmente a la disponibilidad de extensas bases de datos para segmentación, tales como *COCO dataset*, *Open Images dataset*, *PASCAL VOC 2012 dataset* y *Cityscapes dataset*, los cuales contiene entre miles y millones de datos etiquetados para diferentes aplicaciones. Fue en este campo donde se desarrolló uno de los hitos más importantes en el campo de la visión por computadora al presentarse las redes neuronales convolucionales en el modelo AlexNet (Krizhevsky et al., 2012) que luego fue modificándose para crear nuevas arquitecturas más eficientes y precisas, tanto en el problema de segmentación como de clasificación de imágenes.

2.6. La Segmentación de Lagunas

En el campo de la teledetección, uno de los enfoques más importantes es el de monitorear los recursos naturales aprovechando la visión a gran escala que proporcionan los sensores satelitales. Entre los recursos más importantes a monitorear se encuentran los cuerpos de agua superficiales, en los que se considera tanto a los océanos, ríos, lagos, lagunas y humedales.

El cambio climático está modificando las dinámicas ambientales y algunos ecosistemas se consideran como centinelas sensibles a estas modificaciones (Adrian et al., 2009). El agua, al ser un recurso fundamental para las sociedades y para los ecosistemas, ha adquirido mayor relevancia en el campo científico, por lo que se estudia su distribución y sus cambios a través del tiempo. Sin embargo, realizar estudios a gran escala que permitan extraer conclusiones requieren de procesos que son demasiado trabajosos y poco automatizados (Chen et al., 2023), por lo que parte de la comunidad científica se enfoca en desarrollar diversas técnicas que permitan automatizar el trabajo de segmentación de cuerpos de agua en las imágenes satelitales.

Las primeras aproximaciones en la solución de este problema se enfocaban en el desarrollo y uso de índices espectrales para diferenciar los cuerpos de agua gracias a sus características espectrales distintas de otros elementos del entorno, aunque este enfoque puede ser altamente preciso, requiere bastante trabajo manual. Recientemente, se están aplicando diferentes estrategias del campo del aprendizaje de máquina como *support vector machine* (SVM) y *random forest* (RF) en el campo del aprendizaje no supervisado, mientras que el uso de redes artificiales y convolucionales se ha extendido en el campo del aprendizaje supervisado.

2.6.1. Enfoques Basados en Índices Espectrales

En la tarea de segmentación de cuerpos de agua, el uso de índices espectrales es una de las metodologías más usadas debido principalmente a su gran precisión. Sin embargo, su principal desventaja es el gran trabajo que requiere para obtener buenos resultados, por lo que analizar gran cantidad de datos resulta un desafío.

A través del tiempo se han desarrollado diversos índices de diferencia normalizados especializados en la detección de cuerpos de agua. Estos son el índice de agua de diferencia normalizada, NDWI (McFeeters, 1996) y el índice de agua de diferencia normalizada modificado, MNDWI (Xu, 2006).

1. NDWI

   Este índice fue desarrollado por McFeeters en 1996 con la finalidad de delinear cuerpos de agua superficiales basado en imágenes satelitales, para ello, utiliza la radiación reflejada en el espectro de la luz infrarroja cercana (*near infrared, NIR*) y la luz visible verde (McFeeters, 1996), mediante la operación de restar las reflectancias de banda del espectro verde con la banda de NIR y dividiéndolo entre su suma, como se describe en la Ecuación 2.1.

$$NDWI = \frac{Verde - NIR}{Verde + NIR} \tag{2.1}$$

2. MNDWI

   Por su parte, este índice fue desarrollado por Hanqiu Xu en 2006 como una mejora del anterior índice. Para su cálculo se reemplaza la banda NIR por la banda del espectro del infrarrojo de onda corta (*Short-wave infrared, SWIR*). Fue desarrollado con la finalidad de resaltar de mejor manera los cuerpos de agua y disminuir el ruido generado por la vegetación, el suelo y áreas urbanizadas. La Ecuación 2.2 representa el cálculo de este índice.

$$MNDWI = \frac{Verde - SWIR}{Verde + SWIR} \tag{2.2}$$

El resultado de operar con los datos de las bandas verde y NIR para NDWI; y verde y SWIR para MNDWI, resulta en una imagen de un solo canal, en escala de grises, que resalta con valores más altos a los cuerpos de agua. Para realizar la segmentación de de estas imágenes se aplica la técnica de binarización, que consiste en reducir la información a dos valores, por ejemplo, 0 y 1, donde 0 representa la clase no agua y 1, la clase agua. Esta separación se hace escogiendo un umbral adecuado para cada imagen, ya que los resultados de estos índices no son siempre consistentes, los valores de reflectancia cambian de acuerdo a variadas condiciones como la geografía de la zona en cuestión, el ángulo de captura de la imagen satelital, así como artefactos generados. Por estos motivos, la elección de un valor umbral es un trabajo meticuloso que requiere de mucho tiempo, sobretodo considerando la extensión de las imágenes satelitales. Para solucionar este problema, se ha utilizado algoritmos para la automática selección de umbrales en las imágenes, tales como el algoritmo de Otsu (Otsu, 1979).

2.7. Aprendizaje Profundo

El aprendizaje profundo es un subcampo del aprendizaje de máquina y se basa en el uso de redes neuronales artificiales para procesar complejos y extensos datos, pudiendo abstraer características intrínsecas de los datos analizados. A lo largo del tiempo se han desarrollado diferentes técnicas y aproximaciones como las redes neuronales artificiales (*artificial neural networks, ANN*), las redes neuronales convolucionales (*convolutional neural networks, CNN*), las redes neuronales recurrentes (*recurrent neural network, RNN*) y, recientemente, los *transformers*. Estos modelos aprovechan los grandes conjuntos de datos, al poder encontrar y entender patrones complejos inherentes a estos datos. Las principales aplicaciones del aprendizaje profundo es el análisis de secuencias de datos temporales, visión por computadora y procesamiento del lenguaje natural (Goodfellow et al., 2016). El aprendizaje profundo puede dividirse en dos categorías, el aprendizaje supervisado y no supervisado.

2.7.1. Aprendizaje Supervisado

En el campo del aprendizaje profundo, se conoce como aprendizaje supervisado a aquellas aplicaciones donde el conjunto de datos para el entreno abarca tanto los datos de entrada así como sus valores objetivo, ambos en formato de vectores (Bishop, 2006). Entonces, para poder realizar un entrenamiento con este enfoque, es preciso conseguir un conjunto de datos etiquetado adecuadamente de acuerdo al objetivo de la investigación, lo cual puede ser una limitación.

Respecto al problema de segmentación de imágenes basado en modelos de aprendizaje profundo, este encaja en el aprendizaje supervisado, por lo que es necesario un conjunto de datos que permita al modelo aprender a identificar qué píxeles de la imagen pertenecen a una clase determinada. Para ello, se requieren dos elementos; a) la imagen original, de la cual el modelo debe aprender a extraer los elementos de interés, y b) las máscaras asociadas, que definen qué regiones de la imagen pertenecen a cada clase, un ejemplo se muestra en la Figura 2.6. Esto permite realizar el aprendizaje supervisado al modelo para que este se ajuste durante el entrenamiento.

Figura 2.6
Elementos requeridos para entrenamiento de modelo de segmentación basado en DL.

a) Imagen original b) Máscara segmentada

En la Figura 2.6 se aprecia un ejemplo hipotético de segmentación de la clase "gato". A la izquierda se encuentra la imagen original, en este caso de tipo RGB, y a la derecha, la segmentación del objeto que forma la máscara de entrenamiento y que determina el objetivo a aprender del modelo entrenado.

2.7.2. Aprendizaje No Supervisado

Por su parte, el aprendizaje no supervisado es un enfoque en el que el conjunto de datos de entrenamiento solo consiste en los datos de entrada, sin estar acompañado de los valores objetivo. Este enfoque busca que el algoritmo aprenda a separar los datos en grupos que tengan características similares, operando similarmente al concepto de agrupamiento (*clustering*) (Bishop, 2006).

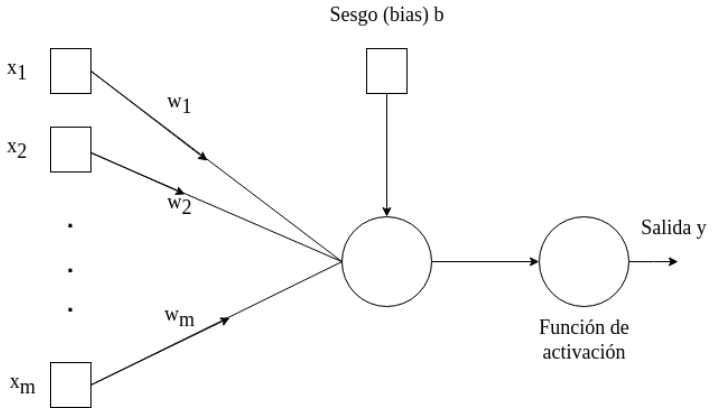
2.8. Redes Neuronales Artificiales y Convolucionales

Las redes neuronales artificiales fueron uno de los primeros enfoques para abordar el aprendizaje profundo. En un principio, estas redes estuvieron inspiradas en el cerebro humano, al identificar algunas características sobre su funcionamiento, como una ser máquina compleja, no lineal y con capacidades de funcionar de forma paralela (Haykin et al., 2009). Por ello, se introdujeron unos elementos básicos llamados perceptrones que intentaban replicar el funcionamiento de las neuronas humanas, estas reciben un valor como entrada, la multiplican por un peso asociado y lo suman con un sesgo (*bias*), algunas veces se les aplica alguna función no lineal como una transformación, conocida como función de activación, que puede ser una función sigmoide, tangente hiperbólica (tanh) o ReLU (rectified linear unit). La salida resultante se utiliza para tomar decisiones o clasificaciones. Los perceptrones pueden agruparse en capas, produciendo lo que se conoce como perceptrón multicapa (*multilayer perceptron, MLP*), el cual es una función matemática que mapea un conjunto de valores de entrada en valores de salida (Goodfellow et al., 2016). En la Figura 2.7, se aprecia el diagrama del funcionamiento básico de un perceptrón, tomando algunos datos de entrada y multiplicándolos por sus pesos correspondientes y sumándolos con un valor b de sesgo.

Matemáticamente, un perceptrón se describe como la suma ponderada de las entradas sumadas con un valor llamado sesgo, para, finalmente, aplicarle una función de activación, esto se muestra en la Ecuación 2.3.

$$y = \sigma(w_1 x_1 + w_2 x_2 + ... + w_m x_m + b), \tag{2.3}$$

Figura 2.7
Funcionamiento de perceptrón, elemento básico de las redes neuronales artificiales.



Donde σ representa la función de activación, generalmente una función no lineal.

Con el desarrollo de la inteligencia artificial, nuevos paradigmas que mejoran el comportamiento de estos sistemas han surgido a través del tiempo. En el campo de la visión por computadora, los principales avances se dieron con la implementación de las redes neuronales convolucionales (CNN). Otras arquitecturas con resultados importantes en el campo, son las redes basadas en mecanismos *transformers* y de atención.

1. Redes neuronales convolucionales (*Convolutional neural network, CNN*)

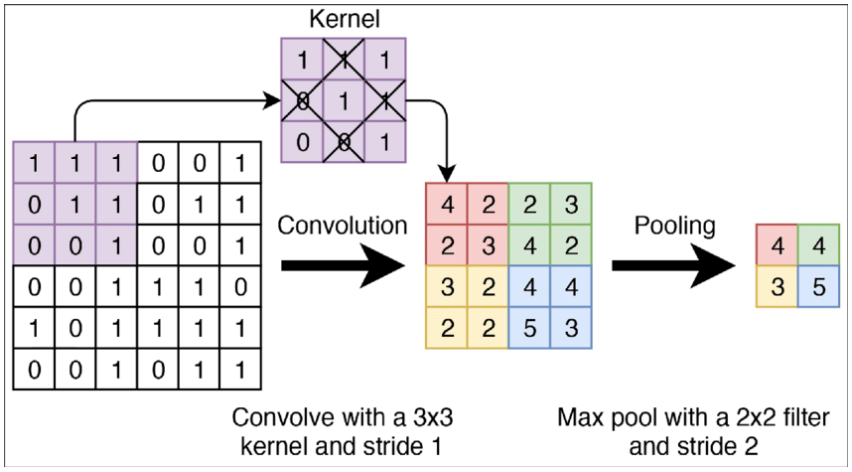
   La característica principal de una CNN es que tiene una topología de tipo grilla. Funcionan tanto para datos en formato de serie temporal, es decir, de una dimensión (1D) como para imágenes, usando grillas de dos dimensiones (2D). Estas redes utilizan la operación matemática de **convolución** combinándola con otra operación llamada *Pooling* (agrupación). En el ámbito de visión por computador, sus principales ventajas son la de capturar información contextual local, pudiendo identificar patrones como bordes, formas y texturas. Para operar la convolución, se precisan de dos funciones: una de ellas es una imagen y la otra función es conocida como kernel en el procesamiento de imágenes, aunque también se le conoce como filtro. La salida de esta operación se le llama mapa de características. Lo que el modelo aprende durante el entrenamiento son los valores del kernel (Goodfellow et al., 2016).

   Por otro lado, la operación de agrupamiento o *pooling* realiza cálculos estadísticos con los valores cercanos o vecinos de cada píxel en una zona específica de la imagen. La salida de

esta operación reemplaza el valor del píxel con el valor calculado estadísticamente. Entre las operaciones de agrupamiento más comunes se tienen el valor máximo (*max pooling*) que obtiene el valor máximo de la ventana de contexto, o el valor promedio (*average pooling*). Estas operaciones permiten reducir las dimensiones de la representación de la imagen.

En la Figura 2.8, se muestra el funcionamiento de la operación de convolución de una imagen 2D con un kernel también 2D y con dimensiones 3×3. Finalmente, al resultado de la convolución se le aplica la función de *maxpoling*, obteniendo, un mapa de características con dimensiones reducidas.

Figura 2.8
Proceso de convolución y agrupamiento.



Nota: Extraído de (Verschoof-Van der Vaart y Lambers, 2019).

2. Otras redes neuronales profundas

   Múltiples arquitecturas han surgido a partir de las CNN, tales como las *Fully convolutional networks*, las redes de tipo UNet y redes generativas adversarias. Variaciones de estas arquitecturas se han planteado para afrontar diferentes problemas de segmentación.

2.9. Algoritmos Auxiliares de Procesamiento

A lo largo de la investigación, se utilizan diversos algoritmos que permiten el procesamiento de la información. Los más importantes son el algoritmo de Canny, la transformada de Hough y el algoritmo de Otsu.

2.9.1. Algoritmo de Canny

Este algoritmo fue propuesto por (Canny, 1986), siendo este uno de los mejores algoritmos detectores de bordes en imágenes, cuyas principales características son: a) una baja tasa de error en la detección de bordes, b) los puntos de borde están bien localizados y c) puede entregar un solo punto como borde.

El algoritmo consiste en la aplicación de una función Gaussiana circular 2-D con la finalidad de suavizar la imagen, la función Gaussiana se expresa en la Ecuación 2.4.

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.4}$$

Sea la imagen descrita mediante $f(x, y)$, la imagen suavizada $f_s(x, y)$ se calcula mediante la convolución con la función Gausiana, como se muestra en la Ecuación 2.5.

$$f_s(x, y) = G(x, y) * f(x, y) \tag{2.5}$$

Donde $*$ representa el operador convolución.

El gradiente de una función es una herramienta matemática que permite encontrar los bordes de la imagen y su dirección para cada punto (x, y) (Gonzales y Woods, 2008). Para una función f de dos variables (x, y), el gradiente se describe mediante la Ecuación 2.6.

$$grad(f) = [g_x \quad g_y]^T = \left[\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y}\right]^T \tag{2.6}$$

La propiedad más importante del gradiente es que apunta en la dirección de la mayor tasa de cambio en el punto (x, y).

Se aplica el gradiente a la imagen suavizada $f_s(x, y)$ y se calcula la magnitud y fase, mediante las Ecuaciones 2.7 y 2.8.

$$M(x, y) = \sqrt{g_x^2 + g_y^2} \tag{2.7}$$

$$\alpha(x, y) = \arctan \frac{g_y}{g_x} \tag{2.8}$$

En base a la fase y magnitud del gradiente se aplica la supresión de no máximos. Para cada punto (x, y) de la imagen procesada, se encuentra la dirección del gradiente más cercano a $\alpha(x, y)$, si la magnitud de $M(x, y)$ es menor a sus vecinos en la dirección de la fase, $g_N(x, y) = 0$, caso contrario $g_N(x, y) = M(x, y)$. Donde $g_N(x, y)$ es la imagen con los no máximos suprimidos.

Para mejorar la detección de bordes, se aplica una doble umbralización (una superior y otra inferior) a $g_N(x, y)$, lo que permite eliminar posibles falsos positivos y negativos. La aplicación de los umbrales se muestra en las Ecuaciones 2.9 y 2.10.

$$g_{Nh}(x, y) = g_N(x, y) \geq T_H \tag{2.9}$$

$$g_{NL}(x, y) = g_N(x, y) \leq T_L \tag{2.10}$$

Se recalcula g_{NL} mediante $g_{NL}(x, y) = g_{NL}(x, y) - g_{NH}(x, y)$. Donde $g_{NH}(x, y)$ representan los píxeles fuertes que son considerados como bordes, mientras que $g_{NL}(x, y)$ son considerados como los píxeles débilmente considerados bordes, los débiles están entre los dos umbrales y solo se conservan si están conectados a bordes fuertes, asegurando que formen parte de un contorno.

En resumen, el algoritmo de Canny es un algoritmo confiable para la detección de bordes, basa su comportamiento en la aplicación de una función gaussiana para suavizar la imagen, la aplicación del gradiente para detectar dónde y hacia qué dirección se producen las tasas de cambio más pronunciadas en la imagen, luego se aplican umbrales para detectar aquellas variaciones que son fuertes y débiles.

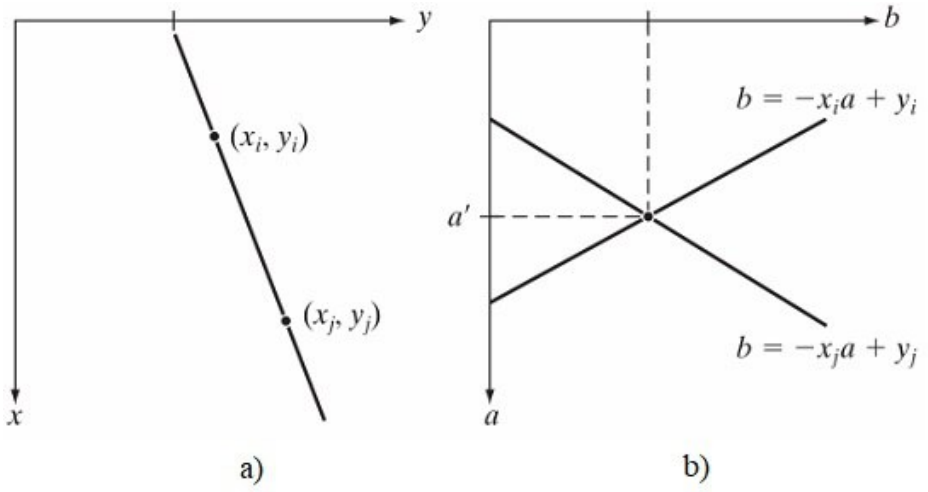
2.9.2. Transformada de Hough

Este algoritmo fue propuesto por Hough (Duda y Hart, 1972) y su utilidad es la de detectar figuras que puedan ser descritas matemáticamente, como rectas, circunferencias o elipses.

Para el caso de detección de rectas, sea (x_i, y_i) un punto en el plano xy y sea la ecuación de la recta definida por $y_i = ax_i + b$. Por el punto (x_i, y_i) pueden pasar infinitas pendientes definidas por los valores de a y b. Hough plantea el espacio de parámetros, considerando el plano ab (los

parámetros que definen una recta) y escribe su ecuación como $b = -x_i a + y_i$, que representa a todos los puntos (x_i, y_i) que cumplen con la ecuación anterior, es decir, que pertenecen a la recta definida por a y b. Esta transformación se muestra en la Figura 2.9.

Figura 2.9
Representación gráfica de la transformada de Hough: a) Plano xy, b) Espacio de parámetros en el plano ab.



Nota: Extraído de (Gonzales y Woods, 2008).

Se observa que dos puntos que pertenecen a una misma recta definida por $y_i = ax_i + b$, en el espacio de parámetros ab forman dos rectas que cuya intersección representa los puntos a y b que definen a la recta en el espacio xy.

Se utiliza un acumulador donde se recopilan las intersecciones entre los diferentes puntos analizados. Aquellos puntos que tienen mayor número de intersecciones son considerados como los que definen las rectas identificadas en la imagen analizada, identificados por a y b.

2.9.3. Algoritmo de Otsu

El problema de umbralización puede ser visto como un problema de decisión estadística. Diferentes algoritmos se han propuesto, siendo Otsu una alternativa atractiva. Este método maximiza la varianza interclase, asumiendo que los píxeles de cada clase (objeto y fondo) tienen intensidades claramente distintas. El cálculo de las varianzas de píxeles se realiza sobre el histograma normalizado de la imagen.

Asumiendo que una imagen digitalizada tiene L intensidades distintas, esto es, $\{0, 1, 2, \ldots, L-1\}$,

se puede calcular el histograma contando cuántos píxeles con determinada intensidad tiene la imagen, esto es, $n_0, n_1, n_2, \ldots, n_{L-1}$. Para normalizar el histograma se divide el número total de píxeles de cada elemento entre el número total de píxeles, $p_i = n_i/MN$. El histograma normalizado tiene L elementos con amplitudes que varían entre 0 y 1.

Se establece un umbral k que pertenece al rango de intensidades y separa la imagen en dos clases, esto se expresa con la Ecuación 2.11.

$$T(k) = k, \quad 0 < k < L - 1 \tag{2.11}$$

El umbral k, separa las clases C_1 (seleccionando los píxeles con intensidades entre $[0, k]$) y C2 (seleccionando los píxeles con intensidades entre $[k + 1, L - 1]$).

Se calcula la suma acumulativa basados en el histograma con la Ecuación 2.12.

$$P_1(k) = \sum_{i=0}^{k} p_i \tag{2.12}$$

Luego, se calcula la intensidad promedio o promedio acumulativo hasta el nivel k, que se expresa como en la Ecuación 2.13.

$$m(k) = \sum_{i=0}^{k} ip_i \tag{2.13}$$

Finalmente, se calcula la intensidad promedio de toda la imagen mediante la Ecuación 2.14.

$$m_G = \sum_{i=0}^{L-1} ip_i \tag{2.14}$$

Con estos resultados previos, se calcula la varianza interclase para cada valor de umbral de $kin[0, L - 1]$. La varianza interclase se define mediante la Ecuación 2.15.

$$\sigma_B^2(k) = \frac{[m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]} \tag{2.15}$$

El valor óptimo de k es aquel que maximiza el valor de la variante interclase de la Ecuación 2.15.

En resumen, el algoritmo de Otsu calcula el histograma normalizado de la imagen con sus

componentes p_i, $\quad i = 0, 1, 2, \ldots, L-1$. Luego calcula la suma acumulativa $P_1(k)$, el promedio acumulativo $m(k)$ y la intensidad promedio global m_G. Con estos valores se calcula la varianza interclase para cada valor de k, obteniendo el valor en el cual k optimiza la máxima variación entre dos clases.

La implementación del algoritmo de Canny, transformada de Hough y el algoritmo de Otsu están implementados en diversos lenguajes como Python y por diversas librerías como OpenCV y Scipy.

2.10. Transformada de Discreta de Fourier en una Dimensión

Sea una función continua en el tiempo definida mediante $f(t)$, se puede obtener la señal muestreada a intervalos ΔT igualmente espaciados mediante la multiplicación de la señal $f(t)$ con la función $sampling$, la cual es un tren de impulsos. La función muestreada se expresa como en la Ecuación 2.16.

$$\hat{f}(t) = f(t)s_{\Delta T} = \sum_{n=-\infty}^{\infty} f(t)\delta(t - n\Delta T) \tag{2.16}$$

Aunque $\hat{f}(t)$ representa a la función $f(t)$ muestreada a intervalos ΔT, esta sigue siendo una función continua. Para obtener la función discreta se utiliza la propiedad de filtrado de la función Delta de Dirac, que se expresa como en la Ecuación 2.17.

$$\int_{-\infty}^{\infty} f(t)\delta(t - t_0)dt = f(t_0) \tag{2.17}$$

Aprovechando esta propiedad, se puede calcular el valor de las muestras de la señal $f(t)$ mediante la integración, como se muestra en la Ecuación 2.18.

$$f_k = \int_{-\infty}^{\infty} f(t)\delta(t - k\Delta T)dt = f(k\Delta T) \tag{2.18}$$

Siendo f_k una señal discreta, para valores de $k = \ldots, -2, -1, 0, 1, 2, \ldots$

A la señal muestreada y continua $\hat{f}(x)$ se le puede calcular la transformada de Fourier, lo que se expresa en la Ecuación 2.19.

$$\hat{F}(\mu) = \int_{-\infty}^{\infty} \hat{f}(t) e^{-j2\pi\mu t} dt \tag{2.19}$$

Reemplazando la Ecuación 2.16 en 2.19, se obtiene que la transformada de Fourier de la señal muestreada es:

$$\hat{F}(\mu) = \sum_{n=-\infty}^{\infty} f_n e^{-j2\pi\mu\Delta T} \tag{2.20}$$

A pesar de que f_n sea una función discreta, $\hat{F}(\mu)$ es una función continua, infinita y periódica con periodo $1/\Delta T$. Ya que $\hat{F}(\mu)$ es periódica, se puede tomar un solo periodo para caracterizarla completamente. Se toman M muestras igualmente espaciadas de un periodo de $\hat{F}(\mu)$, esto se muestra en la Ecuación 2.21.

$$\mu = \frac{m}{M\Delta T} \quad m = 0, 1, 2, \ldots, M-1 \tag{2.21}$$

Sustituyendo la Ecuación 2.21 en 2.20, se obtiene:

$$F_m = \sum_{n=0}^{M-1} f_n e^{-j2\pi mn/M} \quad m = 0, 1, 2, \ldots, M-1 \tag{2.22}$$

La Ecuación 2.22 representa la transformada de Fourier discreta (DFT) para un conjunto de M muestras de $f(t)$. Similarmente, se puede calcular la transformada de Fourier discreta inversa (IDFT) con la Ecuación 2.23.

$$f_n = \frac{1}{M} \sum_{m=0}^{M-1} F_m e^{j2\pi mn/M} \quad n = 0, 1, 2, \ldots, M-1 \tag{2.23}$$

Que recupera la señal discreta en base a las M muestras de la DFT. Las Ecuaciones 2.22 y 2.23, representan el par de transformadas de Fourier discretas.

Todas las demostraciones mostradas en este apartado son extensibles a funciones de dos variables independientes, como lo son las imágenes.

2.11. Pruebas Estadísticas

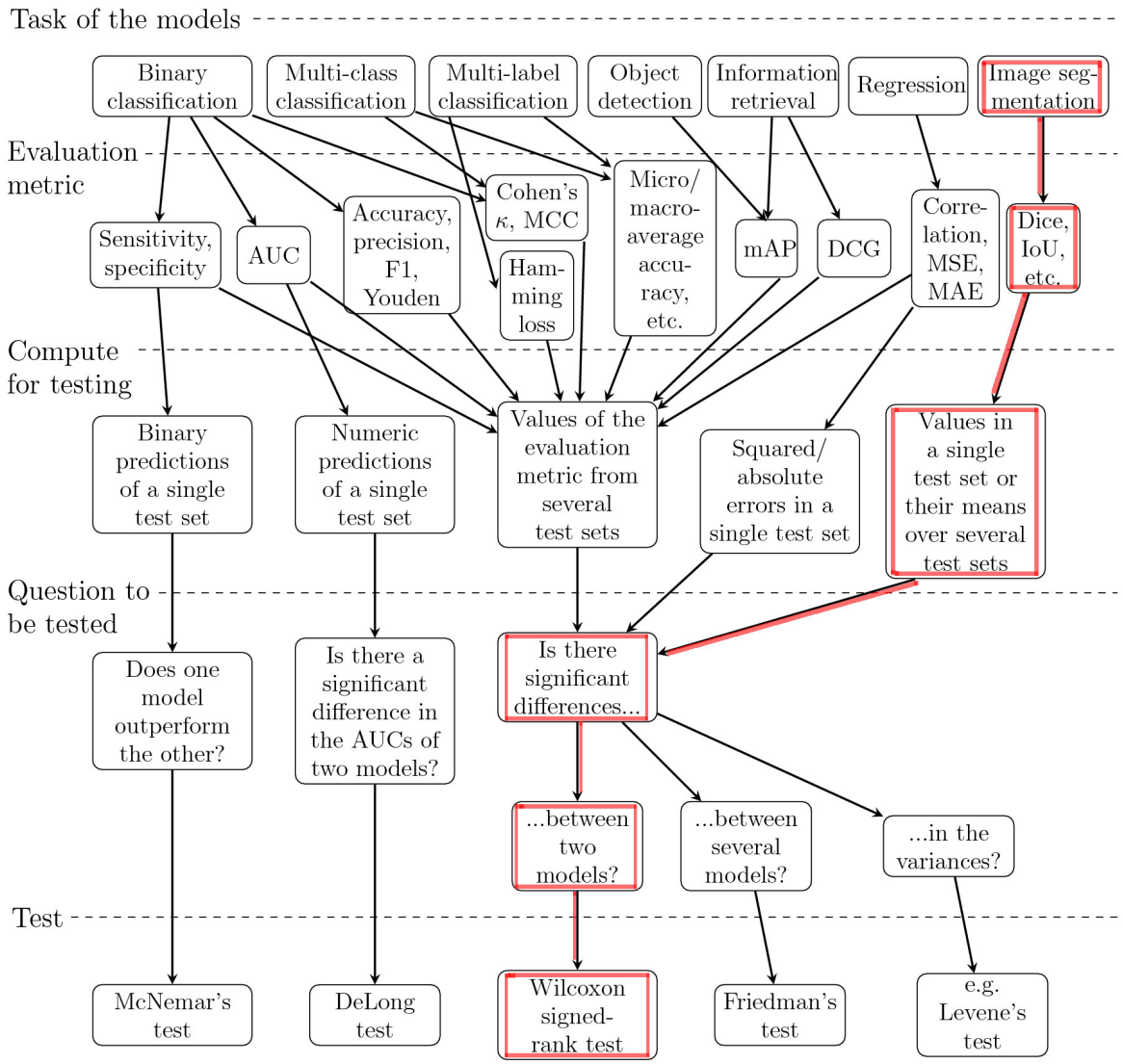
La evaluación de rendimiento y comparación entre modelos es una tarea que se sustenta en análisis estadísticos. Para la tarea de segmentación de imágenes en el paradigma del aprendizaje de máquina supervisado, (Rainio et al., 2024) sugieren diferentes análisis estadísticos para cada tipo de tarea. En la presente investigación se realiza una tarea de segmentación semántica sobre un solo conjunto de datos elaborado específicamente para la zona de estudio. La propuesta de la investigación es la modificación de un modelo base que mejore en la detección de lagunas. De acuerdo a la Figura 2.10, que muestra las diferentes tareas comunes de visión por computadora, las métricas de evaluación comúnmente utilizadas, la pregunta de a la que responde la prueba estadística, y la prueba estadística sugerida, la investigación se ajusta al flujo mostrado en color rojo.

En la presente investigación, la tarea de los diferentes modelos entrenados es la segmentación semántica de lagunas en la cordillera del Vilcanota. Para evaluar su rendimiento se calculan diferentes métricas como MIoU (*Mean Intersection over Union*), *F1-Score* y *Pixel Accuracy*. Dado que el conjunto de datos es realizado específicamente para el estudio, solo se tiene un conjunto de datos. Se espera que la modificación añadida al modelo línea de base mejore su rendimiento. Asumiendo que existe una mejora en las métricas medidas entre los dos modelos, se desea saber si esta mejora (diferencia) es estadísticamente significativa o se debe al azar. En este contexto, siguiendo la Figura 2.10, sugiere la aplicación de la prueba de rangos con signos de Wilcoxon.

2.11.1. Prueba de Rangos con Signos de Wilcoxon

(Rainio et al., 2024) indican que esta es una prueba no paramétrica, es decir, no asume una distribución normal en la diferencia de dos muestras pareadas. Se asume que existe una muestra que es evaluada por dos métodos distintos, por lo que se obtienen dos resultados (pareados porque provienen de la misma muestra). La prueba de Wilcoxon establece la hipótesis nula que la mediana de las diferencias en los pares emparejados es 0. Su estadístico asociado se define como en la Ecuación 2.24.

Figura 2.10
Guía para selección de prueba estadística.



Nota: Se muestran las posibles tareas para un modelo, sus métricas de evaluación, los valores de las métricas de evaluación que deben calcularse para cada modelo antes de la prueba estadística, las preguntas potenciales que una prueba estadística podría responder y la prueba adecuada. Extraído de (Rainio et al., 2024).

$$T = \text{mín}\left\{R^+, R^-\right\}, \quad \text{donde} \quad R^+ = \sum_{d_i>0} rank(d_i), \quad R^- = \sum_{d_i<0} rank(d_i) \tag{2.24}$$

Donde $rank(d_i), \quad i = 0, 1, \ldots, n$ representan las diferencias d_i de las n muestras ordenadas de acuerdo a sus valores absolutos. La significancia estadística del estadístico T se puede calcular usando la tabla de valores críticos para la prueba de Wilcoxon o, si el número de muestras

pareadas es grande, usando el estadístico Z, que, asume una distribución normal en los rangos calculados, este estadístico se muestra en la Ecuación 2.25.

$$Z = \frac{T - n(n+1)/4}{\sqrt{n(n+1)(2n+1)/24}} \tag{2.25}$$

Si el estadístico Z es menor que el valor crítico asociado al nivel de significando seleccionado, se rechaza la hipótesis nula.

Capítulo 3

Desarrollo e Implementación del Marco de Trabajo

Para abordar los problemas descritos en la presente tesis y lograr los objetivos propuestos se requieren diferentes elementos que en conjunto permiten la segmentación precisa de lagunas en la cordillera del Vilcanota, el conjunto de elementos y materiales desarrollados en esta sección se considera como el marco de trabajo sobre el cual se realizarán los experimentos. Dado que el enfoque propuesto está enmarcado dentro del aprendizaje profundo, un aspecto fundamental a tomar en cuenta son los datos que se suministran al modelo planteado, por ello, se detalla todo el proceso de adquisición y procesamiento de la información. El otro punto importante es el desarrollo e implementación del modelo propuesto, donde se indican las características principales que permiten la solución de la tarea de segmentación. También se detallan los métodos que se utilizan como comparación.

3.1. Adquisición y Preparación de los Datos

En esta sección se detallan todos los procesos llevados a cabo para la adquisición, selección, manejo, preprocesamiento y preparación de información para generar el conjunto de datos, insumo indispensable para el entrenamiento y comparación de rendimiento de las diferentes metodologías evaluadas. Para ello, se sigue el diagrama de flujo de la Figura 1.1.

3.1.1. Fuentes de Datos

En el campo de la teledetección, una de las fuentes principales de información son las imágenes satelitales. Estas poseen diferentes características dependiendo del sensor del cual proviene su medición. Se tienen, por ejemplo, imágenes producidas por sistemas de radar y otras que representan información en el espectro visible e infrarrojo.

Como se aprecia en la sección 2.4.4, existen múltiples satélites que recolectan imágenes satelitales a gran escala. Algunos de estos recursos están disponibles gratuitamente desde sus respectivas plataformas, mientras que otros son de acceso limitado o son simplemente inaccesibles. Los satélites actuales están equipados con sensores para capturar diferentes porciones del espectro electromagnético, considerando las bandas del rojo (R), verde (G) y azul (B) visible, así como las bandas de infrarrojo cercano (NIR) y el infrarrojo de onda corta (SWIR). Prácticamente todos estos satélites son capaces de capturar estas informaciones, sin embargo, lo que los diferencia principalmente es su resolución espacial y temporal. De acuerdo a la información de la Tabla 2.2, se aprecia que existen diferencias significativas en la resolución espacial de los diferentes satélites. Las imágenes que tienen menores resoluciones espaciales normalmente son más adecuadas para realizar análisis de objetos o recursos que tienen grandes dimensiones y abarcan campos visuales más amplios, mientas que imágenes con una mayor resolución espacial pueden capturar información detallada pero abarcando un campo visual más reducido.

Para la teledetección de lagunas normalmente se suelen utilizar imágenes que poseen una alta resolución espacial (Y. Li et al., 2022), ya que estos cuerpos de agua no suelen ser demasiado grandes y para una adecuada identificación se requiere suficiente detalle en las imágenes. Otro punto importante para la selección de la fuente de datos está referido al problema que se desea solucionar: respecto a la tarea de monitoreo y seguimiento de lagunas, (Johnes et al., 1994) indican que para analizar la situación de las lagunas tiene que tomarse en cuenta la evolución histórica de estas. En este contexto, resaltan las misiones Landsat que lanzaron diferentes satélites desde 1972, consiguiendo información de la Tierra durante más de 40 años. Se han lanzado hasta la actualidad 9 misiones, siendo la misión Landsat-5 la más larga, estando en funcionamiento desde 1984 hasta 2013. La siguiente más larga es la misión Landsat-7 que ha

recolectado información desde 1999 hasta 2022.

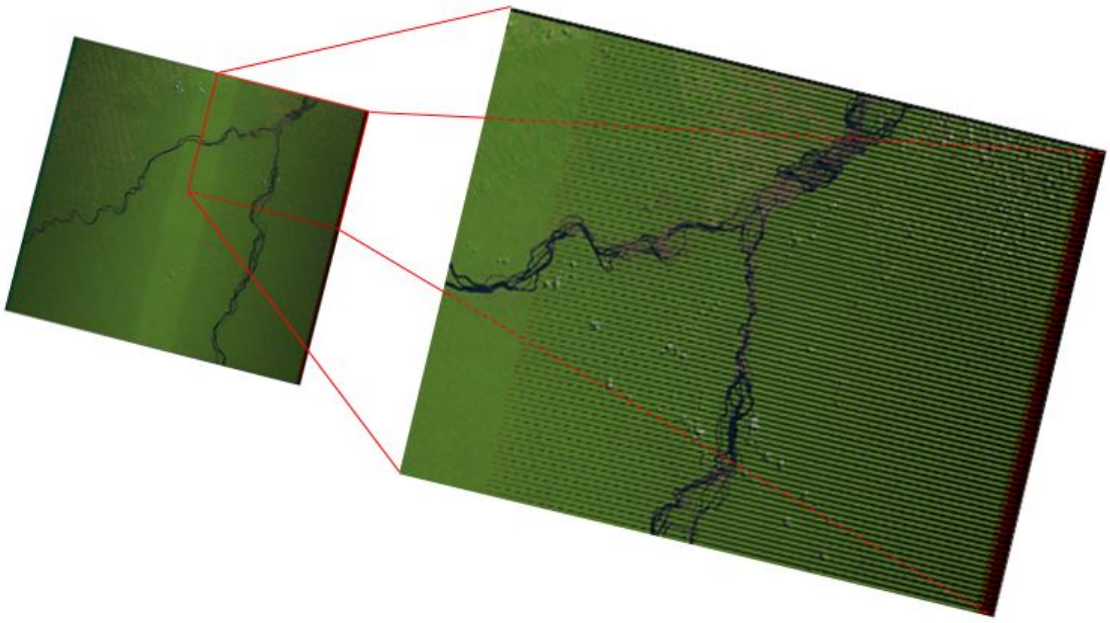
Muchas investigaciones sobre detección de lagunas han abordado el problema mediante la utilización de imágenes RGB, como la realizada por (Garcia et al., 2023), sin embargo, solo utilizar bandas visibles tiene limitaciones debido principalmente a que los cuerpos de agua tienen variadas características de color que pueden ser confundidas con otros elementos del entorno, especialmente en contextos tan complicados las cordilleras andinas. Por ello, otras investigaciones, como la realizada por (Luo et al., 2021), aprovechan la disponibilidad de información de otras bandas espectrales como NIR y SWIR, cuya reflectancia para el agua tiene determinadas características que permiten diferenciar de mejor manera este recurso natural.

Dado que la presente investigación pretende desarrollar un modelo de aprendizaje profundo que pueda servir para realizar un seguimiento cambio de superficie de lagunas a través del tiempo, se escoge a las misiones Landsat como la fuente de datos, ya que cumple las condiciones de una gran ventana temporal de datos, datos multiespectrales y además que son de acceso abierto.

Como se ha mencionado, existen múltiples misiones Landsat, sin embargo, las primeras de ellas no poseían sensores que capturaran bandas espectrales como la NIR y SWIR. Respecto a la misión Landsat-5, esta posee una cuantización de 8 bits para caracterizar los niveles de reflectancia, es decir, tiene poca profundidad de bit. Por otro lado, la misión Landsat-7 ha capturado imágenes multiespectrales durante un periodo bastante extenso, pero tiene una falla conocida: a partir del año 2003, producto de una falla en el sistema de corrección de línea del satélite, las imágenes obtenidas tienen un patrón de zigzag con datos ausentes, tal como se muestra en la Figura 3.1. Según la información de USGS, con este problema solo se conserva el 78 % de los píxeles (USGS, 2024a), por este motivo, para la tarea de segmentación de lagunas este recurso no es el adecuado.

Ya que estas limitaciones son conocidas, múltiples investigaciones de este campo, como las de (Erdem et al., 2021; Liu et al., 2023; Sharma et al., 2024) utilizan la información de la misión Landsat-8, la cual fue lanzada en 2013 y aún en 2024 está en funcionamiento. Las imágenes recopiladas por este satélite abarcan el espectro visible, NIR, SWIR y adicionalmente una banda térmica. Además, este satélite captura imágenes de alta resolución espacial, ya que

Figura 3.1
Ejemplo de escena con patrón de zigzag de Landsat-7.



Nota: Extraído de (USGS, 2024a)

cada píxel registrado representa un cuadrado de 30 metros de lado (900 m^2 de área). Aunque existen otros satélites que proporcionan mejores resoluciones espaciales, como Sentinel-2, estos poseen menos tiempo en órbita.

Por estos motivos, para la presente investigación se utiliza como fuente de datos a los provenientes de la misión **Landsat-8**, ya que es una fuente abierta de imágenes satelitales (disponibles gratuitamente desde la plataforma de USGS, https://earthexplorer.usgs.gov/), registra información multiespectral suficiente para el estudio, sus imágenes tienen alta resolución y han capturado información terrestre durante 11 años.

3.1.2. Características de la Fuente de Datos

El USGS, entidad que administra los datos obtenidos por las misiones Landsat, organizó sus productos en dos colecciones, a saber: a) Colección 1 y b) Colección 2. La primera colección preprocesa las imágenes crudas que el satélite obtiene y asegura una calidad consistente, mientras que la colección 2, basado en los productos anteriores de la colección 1, provee recursos con una precisión de geolocalización absoluta mejorada, además de entregar información de reflectancia,

temperatura superficial y modelos de elevación digital. Debido a que la colección 2 presenta información con más procesamiento y con más precisión respecto a la geolocalización, se escoge esta colección.

Todos los productos entregados por la colección 2 tienen las características comunes detalladas en la Tabla 3.1.

Tabla 3.1
Características de escenas de la Colección 2 de Landsat.

Característica	**Descripción**
Formato de la escena	GeoTIFF (extensión ".TIF").
Tamaño de escena	Aprox. 7000x7000 píxeles.
Tamaño de píxel	30 y 60 metros.
Proyección de mapa	*Universal Transverse Mercator* (UTM).
Orientación de la escena	Con el norte arriba.

Nota: Tabla obtenida de (USGS, 2024b).

A su vez, la colección 2 proporciona dos niveles de preprocesamiento: a) Nivel-1 (*Level-1*) y b) Nivel-2 (*Level-2*), cada uno de ellos con sus propios niveles de procesamiento, los que se muestran a continuación:

1. *Level-1*:
    - *Terrain Precision Correction* (L1TP)
    - *Systematic Terrain Correction* (L1GP)
    - *Geometric Systematic Correction* (L1GS)
2. *Level-2*:
    - *Science Product* (L2SP)
    - *Surface Reflectance* (L2SR)

Respecto a los productos de nivel-1, sus características de detallan en la Tabla 3.2.

Se observa en la Tabla 3.2 que los productos del nivel L1GS son los menos procesados, solo siendo calibrados radiométricamente y con corrección geométrica básica. Los otros dos niveles tienen más procesamiento tanto en su calibración radiométrica como en sus correcciones geométricas, en especial el nivel L1TP, cuyas correcciones eliminan las distorsiones debido a

Tabla 3.2
Niveles de procesamiento para los datos de Colección 2, Nivel-1.

Nivel de Procesamiento	Descripción
Terrain Precision Correction (L1TP)	Calibrado radiométricamente y ortorrectificado usando puntos de control.
Systematic Terrain Correction (L1GP)	Calibrado radiométricamente, con correcciones geométricas sistemáticas y modelos de elevación digital para corrección de desplazamiento de relieve.
Geometric Systematic Correction (L1GS)	Calibrado radiométricamente solo con correcciones geométricas sistemáticas.

Nota: Tabla modificada de (USGS, 2024b)

la topografía del terreno y la altitud del satélite, siendo este el producto de más alta calidad para el nivel-1 de procesamiento.

Con respecto al Nivel-2, estos son generados en base a los productos L1TP (siempre que estén disponibles) o L1GP. Las características de procesamiento de los productos de Nivel-2 se muestran en la Tabla 3.3.

Tabla 3.3
Niveles de procesamiento para los datos de Colección 2, Nivel-2.

Nivel de Procesamiento	Descripción
Science Product (L2SP)	Incluye datos de Reflectancia Superficial (*Surface Reflectance*, SR), Temperatura Superficial (*Surface Temperature*, ST), archivo de coeficientes de ángulos y bandas de calidad.
Surface Reflectance (L2SR)	Incluye datos de Reflectancia Superficial (*Surface Reflectance*, SR), archivo de coeficientes de ángulos y bandas de calidad.

Nota: Tabla modificada de (USGS, 2022)

La diferencia principal entre L2SP y L2SR, es que el producto L2SP contiene tanto los datos de reflectancia superficial (SR) como los de temperatura superficial (ST). Dado que la presente tesis utilizará la información SR multiespectral, cualquiera de los dos productos ofrecidos son válidos como fuente de datos.

El satélite Landsat-8, el cual está equipado con los sensores OLI (*Operational Land Imager*) y TIRS (*Thermal Infrared Sensor*), recoge información multiespectral con el sensor OLI mientras que con el sensor TIRS capta información térmica infrarroja. Para el caso de estudio, interesan los datos entregados por el sensor OLI, cuyas bandas espectrales recolectadas se muestran en la Tabla 3.4.

Tabla 3.4
Especificaciones de las bandas espectrales del sensor OLI.

Banda	Nombre	ID	Unidad	Long. de Onda (μm)
1	Aerosol costero	SR_B1	Sin unidad	0.435 - 0.451
2	Azul	SR_B2	Sin unidad	0.452 - 0.512
3	Verde	SR_B3	Sin unidad	0.533 - 0.590
4	Rojo	SR_B4	Sin unidad	0.636 - 0.673
5	Infrarrojo cercano (NIR)	SR_B5	Sin unidad	0.851 - 0.879
6	Infrarrojo de onda corta (SWIR1)	SR_B6	Sin unidad	1.566 - 1.651
7	Infrarrojo de onda corta (SWIR2)	SR_B7	Sin unidad	2.107 - 2.294

Nota: Tabla modificada de (USGS, 2022).

Los datos de reflectancia superficial son adimensionales, debido a que representan el ratio de la reflectancia reflejada y la incidente, teniendo ambas las mismas unidades. La SR tiene, entonces, un valor que se encuentra entre 0 y 1, considerándose 0 cuando no se refleja radiación y 1 cuando toda la radiación incidente es reflejada. Los datos de SR entregados por la colección 2, nivel-2 tienen las características mostradas en la Tabla 3.5.

Tabla 3.5
Características de los datos de reflectancia de la colección 2, nivel-2.

Característica	Descripción
Tipo de dato	Entero sin signo de 16 bits.
Rango válido	1 - 65455
Valores de escalamiento	Factor de escala: 0.0000275; Compensación aditiva: - 0.2
Valor de relleno	0

Nota: Tabla modificada de (USGS, 2024c)

De la Tabla 3.5, se observa que USGS entrega los datos de SR en tipo de dato entero de 16 bits de profundidad. Esto se debe a que ofrecer la información en datos flotantes (esto es, entre 0 y 1, [0 - 1]) es muy costoso en términos de almacenamiento (ya que un valor flotante ocupa al menos 32 bits de memoria), por ello, estos datos flotantes son reconvertidos a números digitales (*digital number*, DN) que ocupan solo 16 bits. USGS ofrece también los datos necesarios para reescalar la información de DN a valores de SR, en la Tabla 3.5, estos datos son reconocidos como Factor de escala (*scale factor*, SF) y compensación aditiva (*additive offset*, AO), siguiendo la Ecuación 3.1.

$$SR = DN * SF + AO \tag{3.1}$$

El valor válido máximo de DN es considerado 65455, el cual, siguiendo la Ecuación 3.1, devuelve un valor de SR igual a 1.60, que es mayor al valor de SR normal máximo de 1. Esto se debe a que el sensor OLI es susceptible de saturación, muchas veces pudiendo registrar valores de reflectancia superiores a 1. Por otro lado, se observa en la Tabla 3.5 el valor de relleno, que entrega píxeles con valor de 0, lo cual significa que no hay datos para esas zonas, con este valor de DN y haciendo la conversión a SR, se obtiene un valor de -0.2, el cual es inválido. Por ello, (USGS, 2024d) indica que el rango válido de DN se restringe a [7273 - 43636], los que producen valores válidos de SR [0 - 0.999].

3.1.3. Selección de la Escena Satelital

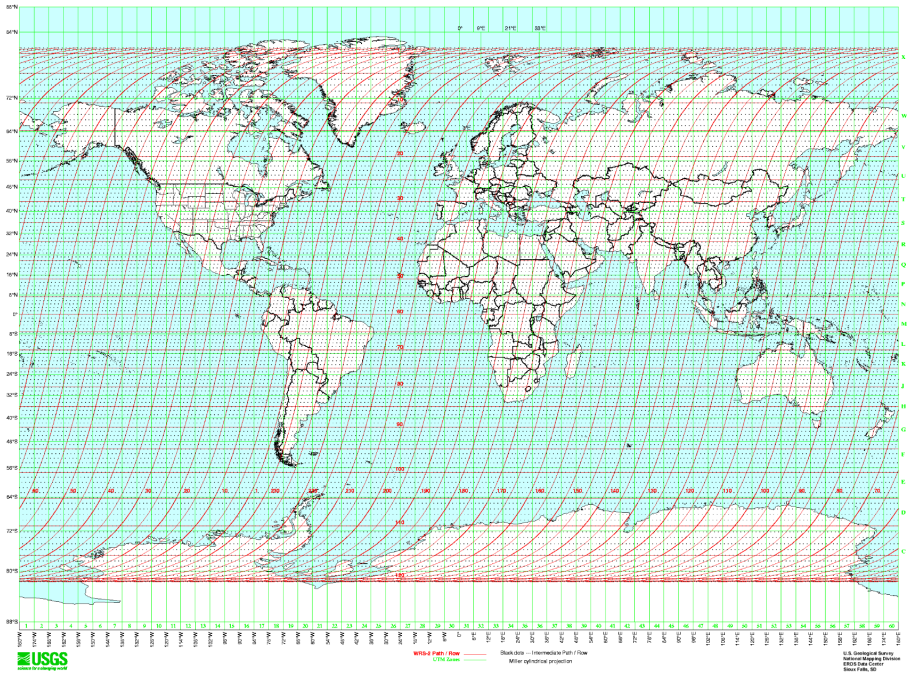
A medida que el satélite sigue su órbita, captura imágenes de la Tierra, cada imagen se denomina escena, estas se ordenan en forma de cuadrícula, siendo indexadas en trayectorias (*PATH*) y filas (*ROW*). (USGS, 2024e) indica que el satélite Landsat-8 sigue la trayectoria definida en el sistema de referencia global - 2 (*world reference system, WRS - 2*), el cual consta en 233 trayectorias de este a oeste, considerando la trayectoria 001 como aquella en la que el satélite cruza por el ecuador a 64.6° de longitud oeste; el valor de *PATH* se refiere a una de estas trayectorias que realiza el satélite. Por otro lado, el término *ROW* se refiere a la componente latitudinal a la que el satélite captura la imagen. Las 233 trayectorias y las filas en que están separadas las escenas que Landsat-8 captura se muestran en la Figura 3.2.

En la Figura 3.2, las líneas verdes representan las separaciones UTM, mientras que las líneas rojas representan las diferentes trayectorias que realiza el satélite. Las líneas punteadas representan las filas en que se captura cada escena.

La presente investigación se centra en la cordillera del Vilcanota, ubicada en el departamento de Cusco, como se aprecia en la Figura 2.1. La escena que captura la cordillera de interés está determinada por el PATH 003 y el ROW 070, esta escena se muestra en la Figura 3.3, la cual contempla la parte sur del Perú con su respectiva cuadrícula de trayectorias y filas. En color amarillo se resalta el departamento de Cusco y el recuadro color rojo identifica la escena **003/070** de la misión Landsat-8 que cubre toda la cordillera del Vilcanota.

En la Figura 3.4 se muestra que la escena seleccionada, en su extremo izquierdo superior,

Figura 3.2
Trayectorias del WRS-2 seguidas por el satélite Landsat-8.



Nota: Extraído de (USGS, 2024e).

contiene a toda la cordillera del Vilcanota.

3.1.4. Criterios para la Generación Conjunto de Datos

Una limitación conocida de las imágenes satelitales es la presencia de nubes y neblina que obstruyen la visibilidad de amplias zonas de las escenas. El Perú tiene dos temporadas claramente diferenciadas, la época de lluvias y la época de secas, por lo que la presencia de nubes y neblina es una limitante principalmente en época de lluvias. Siguiendo la metodología de (Wood et al., 2021), que realizaron un análisis de lagos glaciares en las cordilleras andinas, para la presente investigación se utilizan imágenes de Landsat que poseen nubosidad menor al 10 %, coincidiendo con las temporada de secas (de abril a noviembre). Para obtener un conjunto de datos consistente, las escenas seleccionadas a través de los años deben ser del mismo rango de meses mientras sea posible.

Entonces, considerando los argumentos expuestos en las secciones 3.1.1, 3.1.2, 3.1.3, y las condiciones del párrafo anterior, los criterios para la generación del conjunto de datos se resumen a continuación:

Figura 3.3

Escena 003/070 que captura la cordillera del Vilcanota.

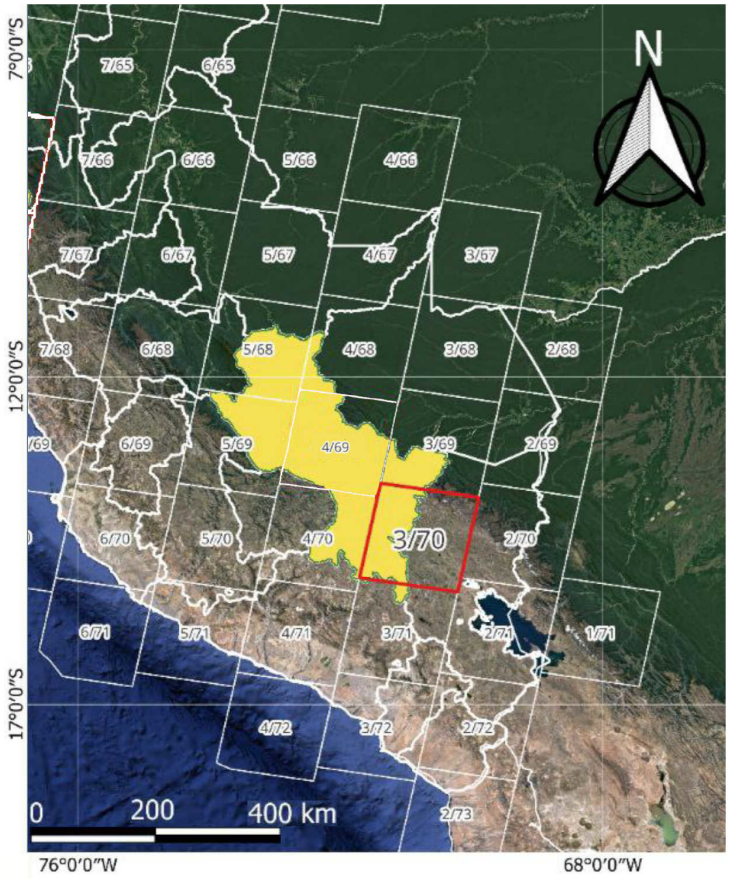


Figura 3.4

Escena 003/070 superpuesta en el mapa.

1. Fuente de datos: Landsat 8.

2. Perteneciente a la colección-2, nivel-2 (C2, L2).

3. Escena satelital: PATH 003, ROW: 070 (003/070).

4. Nivel de procesamiento: L2SP o L2SR.

5. Nivel de nubosidad por escena: menor al 10 %.

6. Meses de captura de datos: entre junio y agosto (para consistencia de datos).

Basados en estos criterios se obtienen las escenas que contienen los datos de SR necesarios para el conjunto de datos durante toda la ventana temporal que el satélite estuvo operativo, esto es, desde 2013 hasta 2023, abarcando 11 años. Se obtiene una escena por año que cumpla los requisitos anteriores, las cuales se detallan en la Tabla 3.6, junto con sus valores de nubosidad.

Tabla 3.6
Escenas seleccionadas para generación del conjunto de datos.

N°	Año	Mes	Nombre de Escena	Proces.	Nub.
1	2013	Julio	LC08_L2SP_003070_20130722_20200912_02_T1	L2SP	5.51 %
2	2014	Junio	LC08_L2SP_003070_20140607_20200911_02_T1	L2SP	1.40 %
3	2015	Junio	LC08_L2SP_003070_20150626_20200909_02_T1	L2SP	0.46 %
4	2016	Junio	LC08_L2SP_003070_20160612_20200906_02_T1	L2SP	5.06 %
5	2017	Agosto	LC08_L2SP_003070_20170802_20200903_02_T1	L2SP	1.74 %
6	2018	Julio	LC08_L2SP_003070_20180704_20200831_02_T1	L2SP	2.61 %
7	2019	Junio	LC08_L2SP_003070_20190621_20200827_02_T1	L2SP	1.33 %
8	2020	Junio	LC08_L2SP_003070_20200623_20200823_02_T1	L2SP	2.50 %
9	2021	Junio	LC08_L2SP_003070_20210610_20210621_02_T1	L2SP	3.23 %
10	2022	Junio	LC08_L2SP_003070_20220629_20220707_02_T1	L2SP	0.64 %
11	2023	Julio	LC08_L2SP_003070_20230702_20230711_02_T1	L2SP	0.29 %

Se observa en la Tabla 3.6 que el nivel de nubosidad en cada una de ellas es menor al 10 %, lo que asegura que la mayor parte de las escenas estén libres de obstrucciones visuales y otros efectos negativos como la neblina. También se aprecia que el nivel de procesamiento de las imágenes es L2SP, conteniendo las escenas con datos de SR así como de ST. Además, la mayoría de las imágenes están capturadas en la misma ventana temporal, esto es, entre junio y julio de cada año en los que el satélite estuvo en órbita.

3.1.5. Generación del Conjunto de Datos

Un aspecto sumamente importante respecto a los conjuntos de datos es su extensión. Es sabido que los modelos de aprendizaje profundo requieren de información abundante. A pesar de que no existe una regla específica para el tamaño de un conjunto de datos, para algunas tareas de segmentación se ha determinado de forma experimental qué cantidades de datos son funcionales y aseguran la capacidad del modelo de generalizar suficientemente bien la información para realizar la tarea en cuestión de forma óptima. En específico, respecto a la tarea de segmentación de lagunas, se han obtenido diferentes bases de conocimiento con diversas cantidades de datos, como se aprecia en la Tabla 3.7.

Tabla 3.7
Tamaño de conjunto de datos para diferentes investigaciones de segmentación de lagunas basadas en DL.

Autores	**Nombre de investigación**	**Tamaño de conjunto de datos**
(Erdem et al., 2021)	An ensemble deep learning based shoreline segmentation approach (WaterNet) from Landsat 8 OLI images	1008
(S. Wang et al., 2022)	A second-order attention network for glacial lake segmentation from remotely sensed imagery	641
(Kaushik et al., 2022)	Automated mapping of glacial lakes using multisource remote sensing data and deep convolutional neural network	660
(Zhao et al., 2023)	Exploring Contrastive Representation for Weakly-Supervised Glacial Lake Extraction	1540

La elaboración de conjuntos de datos para la tarea de segmentación es un trabajo laborioso, en especial para tareas no tan convencionales como la segmentación de lagunas, ya que no existen muchas bases de datos realizadas y de acceso abierto para la tarea en cuestión, además que las características de diferentes locaciones no son necesariamente similares a otras zonas, por lo que la elaboración de bases de datos personalizadas para cada zona de estudio es necesario. En este contexto, la Tabla 3.7 muestra cómo diferentes investigaciones elaboraron conjuntos de datos, conteniendo aproximadamente entre 600 a 1600 imágenes. Para la elaboración del conjunto de datos para la presente investigación se considera estos números como referenciales, ya que se disponen de 11 escenas satelitales de las cuales se debe generar las imágenes para entrenamiento, validación y evaluación de los modelos.

En la presente tesis, basados en la metodología de (Luo et al., 2021), que utiliza 6 bandas del satélite Landsat-8, a saber: SR_B2, SR_B3, SR_B4, SR_B5, SR_B6 y SR_B7 (referidas en la Tabla 3.4); y en la metodología de (S. Wang et al., 2022), que corta las escenas satelitales en pequeños parches de 256x256 píxeles para un rápido procesamiento, se toman en cuenta los siguientes parámetros que definen las características del conjunto de datos:

- Tamaño de conjunto de datos: Entre 600 y 1600 imágenes.
- Resolución de imágenes o parches: 256x256 píxeles.
- Bandas utilizadas: SR_B2, SR_B3, SR_B4, SR_B5, SR_B6 y SR_B7 de Landsat-8.

La elaboración del conjunto de datos consta de 5 procesos: a) combinación de bandas, b) giro de escenas , c) creación de parches, d) selección de parches y e) creación de máscaras; procesos que se detallan a continuación.

Combinación de Bandas

La forma en que USGS entrega las imágenes satelitales es en archivos separados para cada banda espectral, las cuales están en formato *.TIF*, proveniente del estándar *Tag Image File Format, TIFF*, que permite almacenar información de imágenes en alta calidad y sin pérdida de datos y, por ello, con poco nivel de compresión. Además de las bandas espectrales, se proporcionan otros metadatos en archivos ".xml", ".txt", ".json", los cuales tienen información relevante acerca de la posición global de la escena, la fecha de adquisición, valores máximos y mínimos de reflectancia, los factores de reescalamiento y compensación aditiva, entre otros valores. La distribución de estos archivos se muestra en la Figura 3.5.

Se observa que cada archivo tiene un peso de alrededor de 100MB, esto debido a que cada escena representa una imagen de 7592x7721 píxeles. Las imágenes tal como están no pueden ser introducidas al modelo, ya que tienen un tamaño muy grande para su procesamiento y además están separadas, por ello, es necesario crear una imagen que contenga la información de todas las bandas juntas. Esto se logra mediante el apilamiento de cada banda espectral una tras de otra, resultando en una imagen con seis canales. El proceso de apilamiento se muestra en la Figura 3.6.

Figura 3.5
Archivos por escena entregados por USGS.

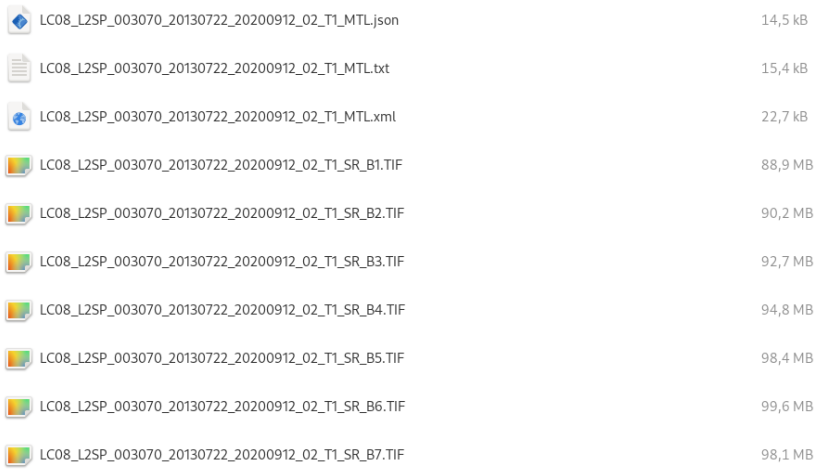
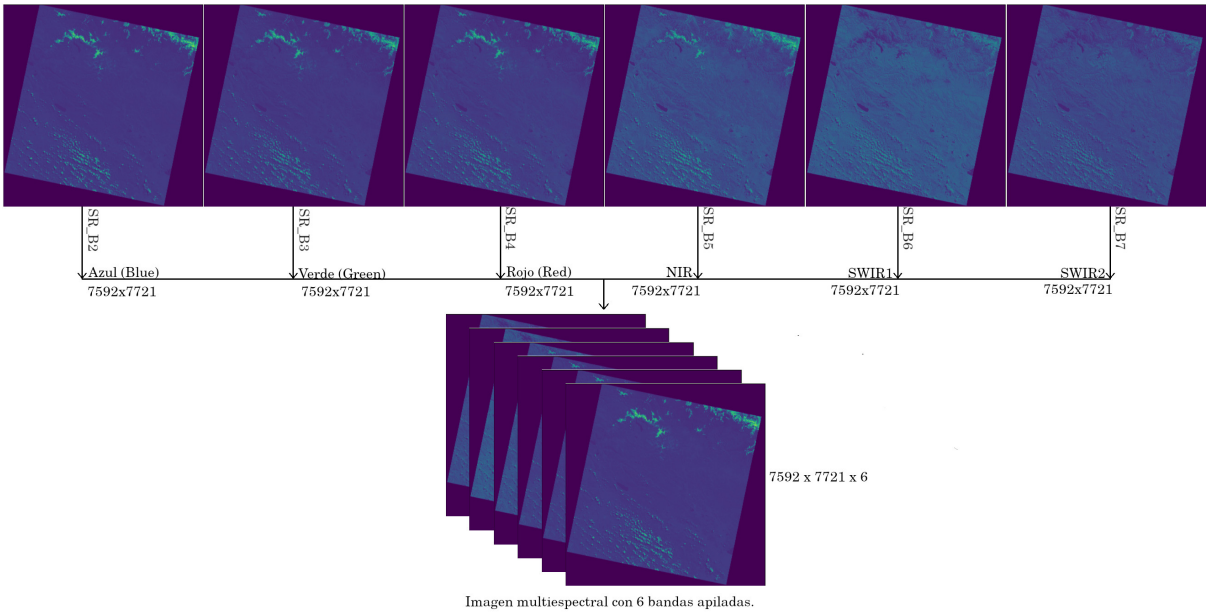


Figura 3.6
Proceso de apilamiento o combinación de bandas.

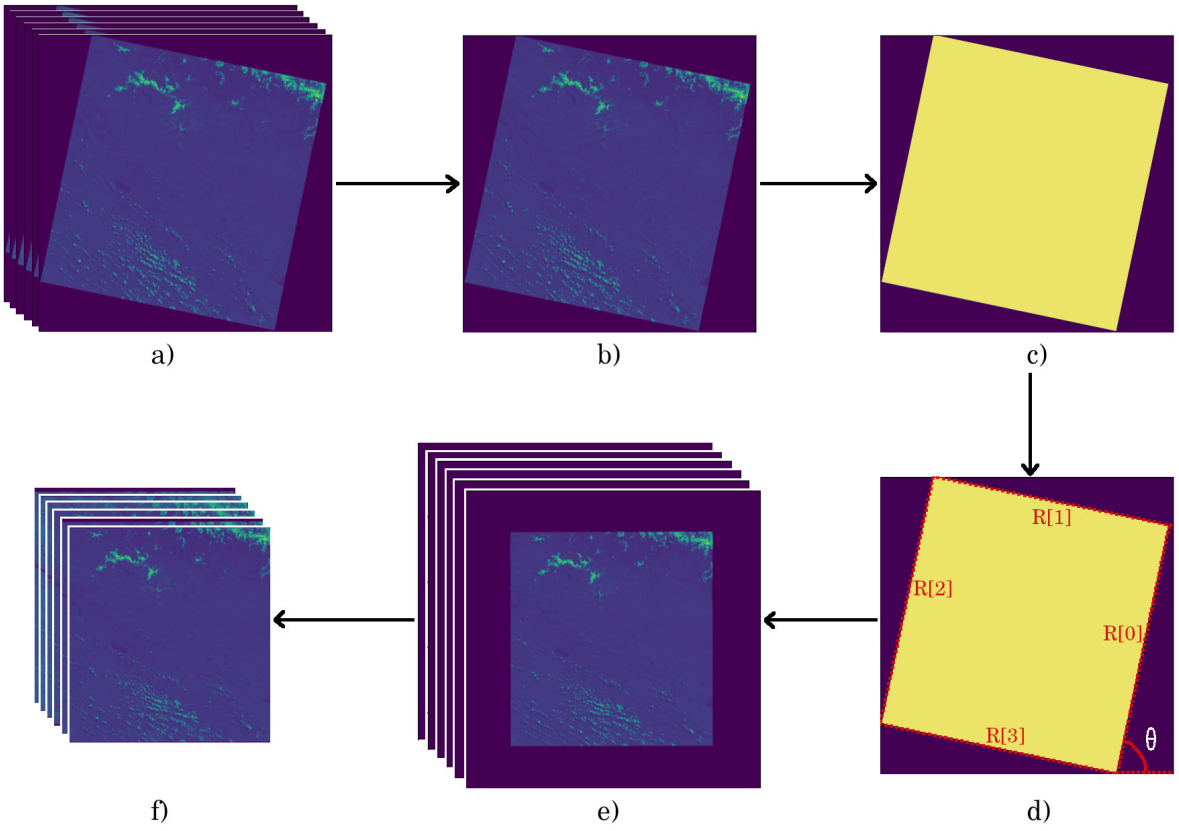


Cada archivo *.TIF* representa una banda específica y tiene dimensiones de 7592x7721 píxeles, representados mediante un tipo entero con profundidad de 16 bits (0 a 65535). Este tipo de archivo se lee con paquetes de software específicos, en este caso se utiliza *rasterio*, un paquete que proporciona una función que permite abrir, guardar y escribir formatos que son utilizados en imágenes satelitales y modelos de terreno. Como se observa en la Figura 3.6, se leen las bandas del espectro visible (B, G, R), las del infrarrojo cercano (NIR) y las bandas de infrarrojo de onda corta (SWIR1 y SWIR2), las cuales se apilan en una sola imagen multiespectral.

Giro de escenas

De acuerdo a la información de la Tabla 3.1 y a la Figura 3.6, la orientación de las escenas es con el norte arriba y debido a la trayectoria del satélite, la escena capturada tiene cierta inclinación, la cual debe ser corregida para poder aprovechar la totalidad de la escena, además, se observa que debido a la inclinación existen bordes que no contienen información, los cuales deben ser eliminados. Dado que todas las escenas comparten el mismo ángulo de inclinación, la determinación de este se realiza solo utilizando una banda espectral. Para ello, se hace uso del algoritmo de Canny y la transformada de Hough, siendo el primero para la detección precisa de bordes y el segundo para la determinación de las rectas que definen los bordes, permitiendo calcular el ángulo de inclinación para corregir la orientación de la escena. El proceso de corrección de inclinación de la escena se muestra en la Figura 3.7.

Figura 3.7
Proceso de corrección de ángulo de inclinación de la imagen de 6 bandas.



Se aprecia en la Figura 3.7 todos los estados y procesos necesarios para la corrección de inclinación. a) representa la escena con 6 bandas espectrales, b) se escoge cualquiera de los

canales ya que todos están alineados, c) se binariza la imagen, transformando cualquier valor mayor a 0 en 255 y manteniendo en 0 lo demás, d) con la imagen binarizada y mediante el algoritmo de Canny, que aplica un filtro Gaussiano para suavizar la imagen, se detectan los bordes de la escena, mostrados en color rojo y mediante la aplicación de la transformada de Hough, se identifican las rectas que definen a los bordes; tomándose en cuenta solo la primera recta, R[0], se determina el ángulo de inclinación θ; e) con el ángulo identificado, se procede a la rotación de la imagen, enderezándola, f) finalmente, se recorta la escena eliminando los bordes sin información. La imagen final resulta con una dimensión de 5888x5888 píxeles y con 6 bandas espectrales. El pseudocódigo para la implementación de estos procesos se muestra a continuación.

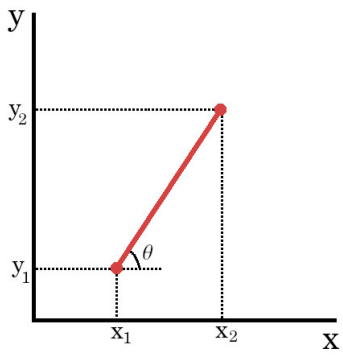
Algoritmo Corrección de inclinación escena satelital de 6 bandas

1: **Entrada:** Imagen satelital de 6 bandas.
2: Selección de una sola banda.
3: Binarización de la imagen.
4: Aplicación de filtro Gaussiano.
5: Aplicación de algoritmo de Canny para detectar bordes.
6: Uso de Transformada de Hough para detección de rectas.
7: Seleccionar la primera recta R[0].
8: Cálculo de la pendiente de la recta θ.
9: Enderezamiento de la imagen según el ángulo θ.
10: Recortar y eliminar los bordes.
11: **Salida:** Imagen enderezada y recortada.

Tanto del algoritmo de Canny como la transformada de Hough están disponibles desde el paquete de software OpenCV. Respecto a la determinación del ángulo θ, la función *HoughLinesP* de OpenCV entrega las rectas identificadas en la escena binarizada, estas se definen a través de los 2 extremos de la recta detectada, definidos como $P(x_1, y_1)$ y $P(x_2, y_2)$. Por ejemplo, se muestra en la Figura 3.8 una recta y los valores que la describen. Para calcular el ángulo θ se determina el arcotangente siguiendo la Ecuación 3.2.

$$\theta = \tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \tag{3.2}$$

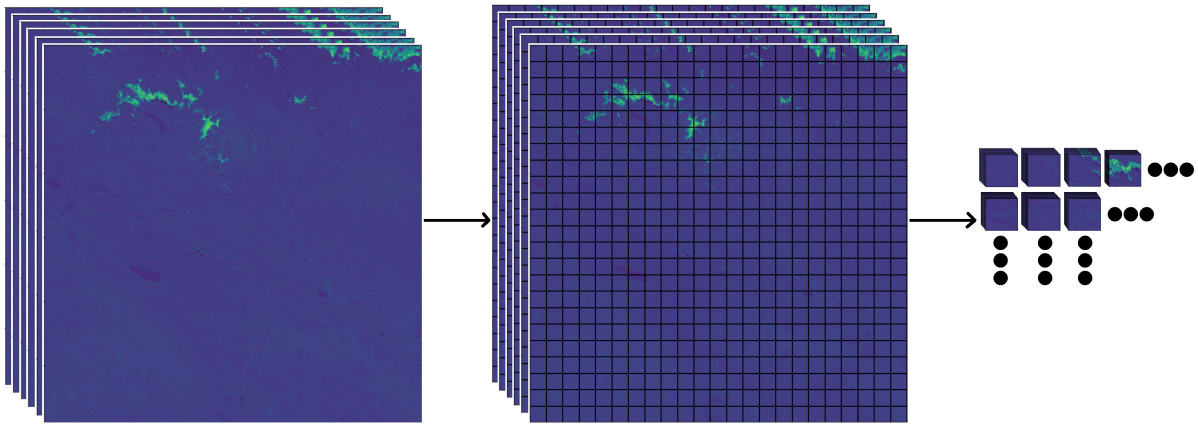
Figura 3.8
Recta definida por los extremos del borde detectado.



Creación de parches

Con la escena de 6 bandas ya enderezada y recortada, se procede a generar los parches o pequeños pedazos de la imagen original. En este caso, cada parche tiene una dimensión de 256x256 píxeles, cada uno de ellos se renombra añadiendo al nombre de la escena un sufijo que consiste en el número de fila y columna al que pertenece. El proceso se observa de forma gráfica en la Figura 3.9.

Figura 3.9
Proceso de creación de parches de 256x256 píxeles.



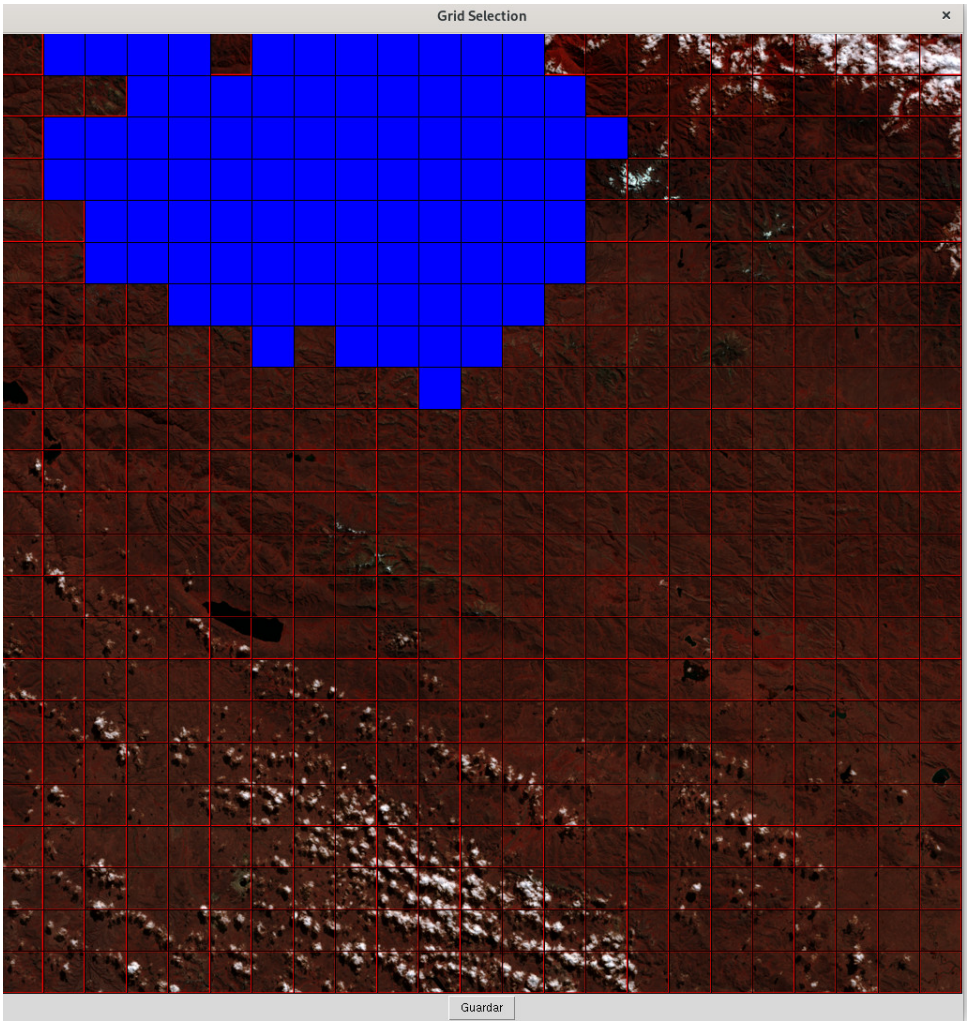
Considerando que la escena recortada tiene unas dimensiones de 5888x5888 píxeles, se obtienen 23 filas y columnas de 256x256 píxeles, resultando en 529 parches por escena.

Los códigos que realizan la lectura de las imágenes *,TIF*, el apilamiento para formar la imagen de 6 bandas, el enderezamiento y recorte, así como la creación de los parches de 256×256 se muestra en el Apéndice A.

Selección de parches

No todos los parches de una escena abarcan la zona de estudio, por ello, se seleccionan solo aquellos parches que pertenezcan a la cordillera del Vilcanota y posean lagunas. Con este fin, se elabora una aplicación gráfica simple basada en el paquete *tkinter* que permite seleccionar los parches de la zona de interés. La interfaz de la aplicación se muestra en la Figura 3.10.

Figura 3.10
Interfaz de aplicación para seleccionar parches de la zona de estudio.



Para facilitar la visualización de cuerpos de agua se utiliza una representación en falso color de la banda NIR, ya que el agua tiende a absorber fuertemente la radiación del infrarrojo cercano, observándose en la representación gráfica de color oscuro respecto de otros elementos, esto se aprecia efectivamente en la Figura 3.10. La aplicación permite seleccionar solo aquellos parches de interés, los cuales se muestran en color azul. Este proceso se realiza para las 11

escenas.

Luego de la selección de parches, se hace una depuración manual de los datos recabados mediante inspección visual de los parches. La cantidad de parches seleccionados y los depurados se muestran en la Tabla 3.8.

Tabla 3.8
Cantidad de parches seleccionados y depurados por escena.

Nombre de Escena	**Parches seleccionados**	**Parches depurados**
LC08_L2SP_003070_20130722_20200912_02_T1	86	72
LC08_L2SP_003070_20140607_20200911_02_T1	85	71
LC08_L2SP_003070_20150626_20200909_02_T1	85	73
LC08_L2SP_003070_20160612_20200906_02_T1	74	68
LC08_L2SP_003070_20170802_20200903_02_T1	82	76
LC08_L2SP_003070_20180704_20200831_02_T1	86	78
LC08_L2SP_003070_20190621_20200827_02_T1	80	73
LC08_L2SP_003070_20200623_20200823_02_T1	78	69
LC08_L2SP_003070_20210610_20210621_02_T1	84	72
LC08_L2SP_003070_20220629_20220707_02_T1	87	76
LC08_L2SP_003070_20230702_20230711_02_T1	81	72
Total	908	800

Después del proceso de depuración, se obtienen 800 parches que cumplen con los requisitos de pertenecer a la cordillera del Vilcanota y poseer lagunas.

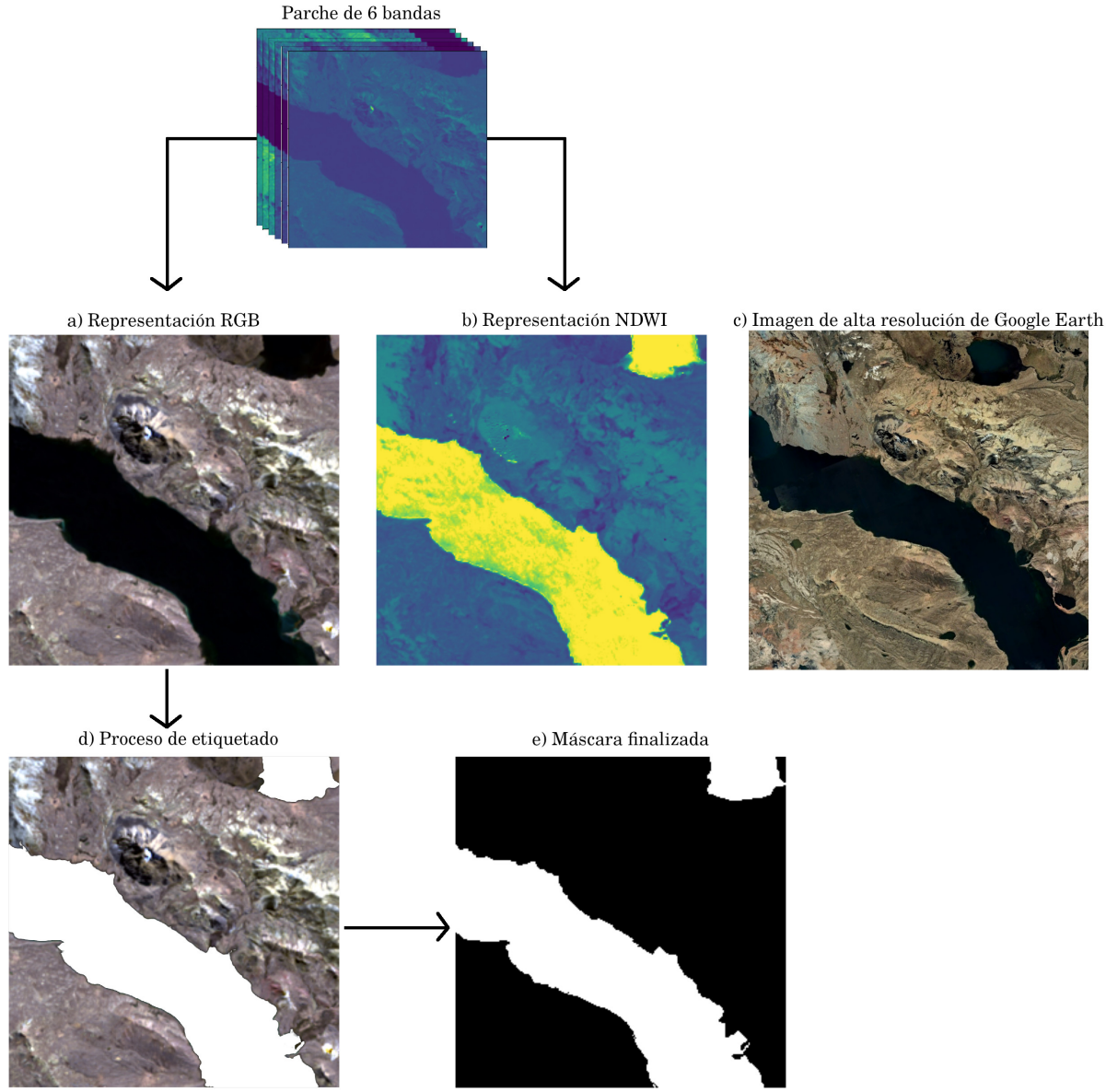
Creación de máscaras

Los 800 parches obtenidos son el material fundamental para la elaboración del conjunto de datos. El otro material indispensable son las máscaras asociadas a cada parche que identifican las regiones donde se ubican las lagunas. La elaboración de máscaras es un proceso manual y meticuloso.

Existen muchas dificultades en la identificación correcta de lagunas, debido principalmente a sus características físicas como diferentes formas, turbidez y coloración, así como a factores ambientales tales como sombras de montañas, nubes y glaciares (Zhao et al., 2023), siendo así difícil identificar las lagunas incluso a simple vista. Por ello, el proceso de generación de las máscaras se realiza con 3 apoyos fundamentales: a) la representación RGB de los parches, b) el NDWI de los parches y c) una imagen de alta resolución, como se muestra en la Figura 3.11.

La elaboración de las máscaras se realizan sobre la representación RGB mediante el uso del software GIMP, un programa de acceso abierto que permite la edición sencilla de imágenes.

Figura 3.11
Proceso de creación de máscaras basado en 3 representaciones.



Nota: Las representaciones son a) RGB, b) NDWI, c) Imagen de alta resolución, d) Etiquetado de lagunas, e) Máscara finalizada.

El parche de 6 bandas no es directamente reproducible por ningún programa, por ello, se extraen las bandas Rojo (R), Verde (G) y Azul (B) para generar la representación RGB, en base a la cual se genera las máscara asociada. Debido a la dificultad en algunos casos de diferenciar a simple vista las lagunas, se utiliza la representación NDWI (en base a las bandas verde y NIR) ya que, en general, detecta adecuadamente cuerpos de agua. En base a estos datos, se

etiquetan los cuerpos de agua sobre la imagen RGB. Finalmente, se binariza la máscara usando dos valores: 255 (blanco) para indicar la clase laguna y 0 (negro) para la clase no-laguna. Se realiza una validación cruzada de la máscara realizada mediante una comprobación visual con la imagen de alta resolución de proveniente de Google Earth.

Finalmente, se obtiene el conjunto de datos, consistente en 800 pares de parches de 6 bandas y sus respectivas máscaras asociadas.

3.1.6. Análisis Exploratorio de los Datos

Cada conjunto de datos tiene sus propias características y limitaciones, por ello se realiza un análisis para poder tomar decisiones respecto al procesamiento de estos datos.

Descripción del Conjunto de Datos

El conjunto de datos elaborado consta de 800 pares de imágenes y máscaras. Sus características generales se muestran en la Tabla 3.9.

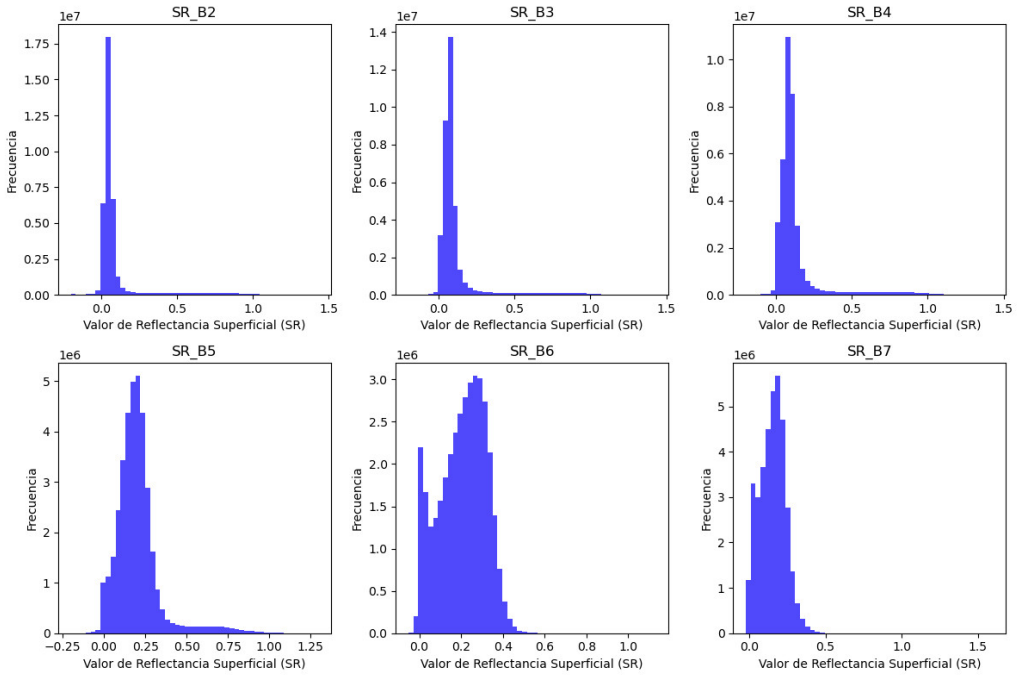
Tabla 3.9
Características generales del conjunto de datos elaborado.

Características	**Imagen**	**Máscara**
Formato de almacenamiento	.TIF	.TIF
Número de canales	6	1
Tipo de dato	Entero sin signo de 16 bits	Entero sin signo de 8 btis
Rango de valores	[0-65535]	[0-255]
Dimensiones	256x256 píxeles	256x256 píxeles

Rango de Valores y Datos Faltantes

Respecto a las imágenes de 6 bandas (parches): de acuerdo a lo mencionado en la sección 3.1.2, los datos de DN deben ser transformados a valores de reflectancia superficial (SR), para ello, se sigue la Ecuación 3.1, transformando el rango [0-65535] a [-0.2, 1.60]. Luego de la transformación, se determinan los histogramas de todo el conjunto de datos (800 imágenes) para cada banda, lo cual se muestra en la Figura 3.12.

Figura 3.12
Histograma de SR de cada banda del conjunto de datos.

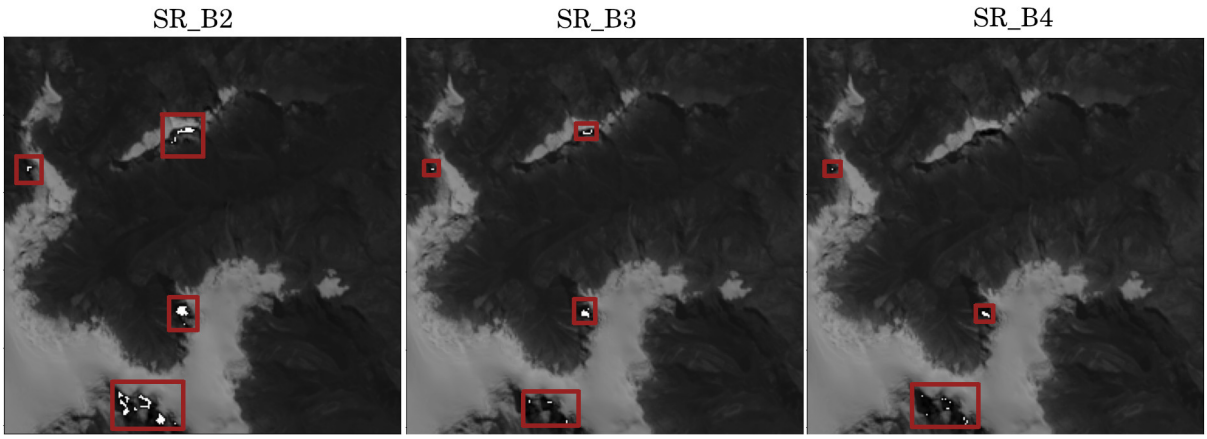


En la Figura 3.12 se observa que en general, la mayor parte de los píxeles se encuentran en el rango entre 0 y 0.5 de reflectancia superficial. Sin embargo, en las bandas SR_B2, SR_B3 y SR_B4 existen valores de SR que superan el límite de 1. Además, en todas las bandas existe un número pequeño de valores de SR negativos, los cuales no son adecuados. Finalmente, se observa que en las bandas SR_B2, SR_B3, SR_B4, SR_B5 y SR_B6 existen valores de reflectancia con valor de $-0{,}2$, los cuales representan el valor de relleno (*fill value*) cuando no se tienen datos, lo que refleja zonas muertas o sin información, esto es posible visualizarlo en la Figura 3.13.

Los valores de relleno en los productos de Landsat, se refieren a aquellos datos que no fueron capturados correctamente por el sensor del satélite o que en el preprocesamiento por USGS fueron descartados por no entregar información confiable.

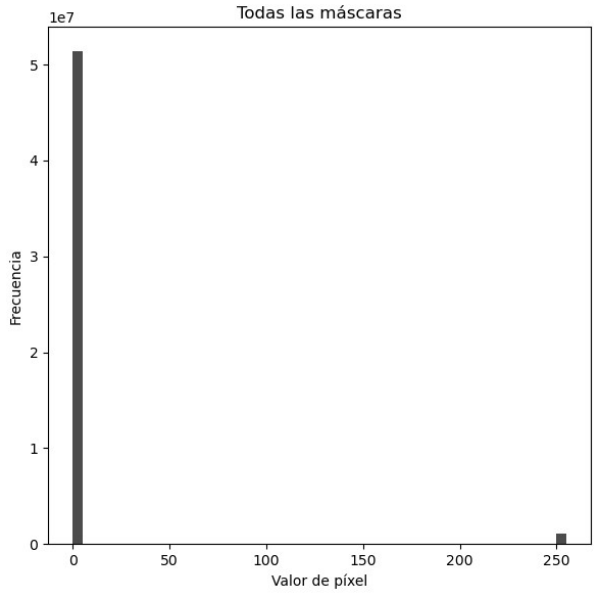
Respecto a las máscaras: estas son de una sola banda y sus valores están binarizados, asumiendo el valor de 0 para representar zonas sin lagunas y el valor de 255 para zonas con presencia de lagunas. Se realiza un análisis de las máscaras contando cuántos píxeles de todas las imágenes representan lagos y cuántos no. En base a ese conteo, se genera el histograma mostrado en la Figura 3.14, donde aprecia que el 97.94 % de los píxeles de las máscaras tienen

Figura 3.13
Ejemplo de datos faltantes en bandas SR_B2, SR_B3 y SR_B4.



valor de 0 que representan no-lagunas mientras que el 2.06 % representa la clase lagunas, siendo la esta la clase no dominante.

Figura 3.14
Histograma de máscaras creadas.



3.1.7. Procesamiento del Conjunto de Datos

Para manejar las limitaciones indicadas en la sección 3.1.6, se realiza el procesamiento de los datos.

Gestión de Datos Faltantes

Según la Tabla 3.3, se observa que el valor de relleno en *digital number*, DN, es 0. Cuando este valor es reescalado a reflectancia superficial, este da un valor de −0,2. Por otro lado, de la Figura 3.12 se advierte la presencia de valores SR negativos, los cuales no son válidos.

Sea $I_c(x, y)$ la imagen correspondiente a los valores de SR en cada píxel (x, y) de un determinado canal c. Se define una nueva imagen $I'_c(x, y)$ de la siguiente manera:

$$I'_c(x, y) = \text{máx}(I_c(x, y), 0) \quad \text{para} \quad c = 1, 2, \ldots, 6 \tag{3.3}$$

Es decir, para cada píxel (x, y), si el valor de $I_c(x, y) < 0$, se asigna 0, caso contrario, se mantiene el valor original. Mediante la operación anterior, se manejan tanto los valores faltantes así como los valores negativos.

Normalización de Datos

Respecto a las imágenes de 6 bandas (parches): Después del manejo de datos faltantes y negativos, el nuevo rango de valores de los parches es $[0, 1.60]$. A pesar de que se manejaron los valores negativos, aún existen valores atípicos en el extremo superior del rango. Como se observa en la Figura 3.12, un número bastante bajo de los píxeles pasa del valor normal máximo de SR (1), por ello, se aplica un preprocesamiento que consiste en reemplazar los datos atípicos con el valor máximo normal. Se hace uso de percentiles que son una medida estadística que separa los datos en 100 partes iguales y permiten la detección de valores extremos, tanto superior como inferior. Este preprocesamiento puede describirse en los siguientes pasos:

Sea $I'_c(x, y)$ la imagen de una banda c con la corrección de datos faltantes, se calculan los percentiles 1 y 99, denotados como p_{c1} y p_{c99}, que separan los valores extremos máximos y mínimos de cada canal del parche de 6 bandas, se puede expresar como en las Ecuaciones 3.4 y 3.5.

$$p_{c1} = Percentil(I'_c, 1) \quad \text{para} \quad c = 1, 2, \ldots, 6 \tag{3.4}$$

$$p_{c99} = Percentil(I'_c, 99) \quad \text{para} \quad c = 1, 2, \ldots, 6 \tag{3.5}$$

Con los percentiles calculados para separar los datos atípicos, y siguiendo el método de normalización para imágenes espectrales recomendado por (Cao et al., 2017), se realiza una normalización lineal de cada canal o llamada normalización Max-Min, escalando los valores de forma lineal para que p_{c1} se mapee a 0 y p_{c99} se mapee a 1. Para ello se sigue la Ecuación 3.6.

$$I'_c = \frac{I'_c - p_{c1}}{p_{c99} - p_{c1}} \quad \text{para} \quad c = 1, 2, \ldots, 6 \tag{3.6}$$

Dado que cada banda se normaliza con los percentiles 1 y 99, los valores de I'_c pueden exceder el valor de 1, por lo tanto, se realiza un recorte haciendo que los valores mayores a 1 se recorten a límite de 1. Con este procesamiento se eliminan los valores extremos atípicos y se normalizan los datos en el rango [0-1].

Respecto a las máscaras:

Las máscaras tienen su valores en tipo de dato entero sin signo de 8 bits [0, 255]. Para poder ser utilizado en los modelos, estos valores deben ser reescalados entre [0 - 1], para ello, simplemente se dividen los valores de los píxeles de la máscara por el valor máximo (255), como se muestra en la Ecuación 3.7. Sea $M(x, y)$ la imagen que representa una máscara, se define $M'(x, y)$ como la máscara normalizada de entre 0 y 1.

$$M'(x, y) = \frac{M(x, y)}{255} \tag{3.7}$$

3.1.8. Separación de Datos

Siguiendo la metodología de (Erdem et al., 2021) para la separación de datos, de forma aleatoria se separa el 70 % de los datos para entrenamiento, 10 % para la validación y 20 % para el evaluación. La aleatoriedad permite asegurar la representatividad de los diferentes tipos de lagos en todos los subconjuntos. La base de datos para los experimentos queda distribuida de acuerdo a la Tabla 3.10.

En la Tabla 3.11, se resumen las características de la base de datos para entrenar, validar y

Tabla 3.10
Separación del conjunto de datos.

Categoría	Cantidad	Porcentaje	Resolución
Entrenamiento	560	70 %	256x256
Validación	80	10 %	256x256
Evaluación	160	20 %	256x256

evaluar las metodologías de segmentación de lagunas.

Tabla 3.11
Resumen de base de datos para entrenamiento, validación y evaluación.

Características	Parches	Máscaras
Número de canales	6	1
Normalización	[0-1]	[0-1]
Tipo de dato	Flotante de 32 bits	Entero de 8 bits
Dimensiones	256x256	256x256
Cantidad	800	800

3.2. Diseño e Implementación del Modelo Propuesto

En esta sección se detalla el proceso de diseño e implementación del modelo propuesto para la tarea de segmentación semántica de lagunas en la cordillera del Vilcanota. Como línea de base para la propuesta se toma la arquitectura UNet, un modelo de segmentación propuesto inicialmente para segmentar imágenes médicas, aunque fue implementado en adelante para muchas otras tareas dando excelentes resultados. La ventaja principal de este modelo es que está optimizado para obtener buenos resultados incluso con pocos datos anotados, lo cual es una limitación en el campo del procesamiento de imágenes satelitales. La modificación realizada al modelo base consiste en la adición de un módulo basado en la aplicación de la transformada rápida de Fourier (*Fast Fourier Transform*, FFT) y la aplicación de filtros en el dominio de la frecuencia en las conexiones de "salto"(*skip*), lo que permite combinar de forma más compleja y con mayor contexto la información del codificador y decodificador.

3.2.1. Vista General del Modelo Línea de Base UNet

UNet es un modelo de segmentación propuesto por (Ronneberger et al., 2015) específicamente para tratar con imágenes médicas, en este caso, segmentación de estructuras neuronales en imágenes producidas por microscopios electrónicos. UNet intentaba principalmente resolver dos problemas de otros modelos anteriores: a) la velocidad de procesamiento de imágenes y b) el uso del contexto de la imagen. Para abordar estos problemas, el modelo UNet se basa en redes completamente convolucionales o FCN, asumiendo una arquitectura codificador-decodificador con forma de U. Con la finalidad de aprovechar la información obtenida en la etapa de codificación se utiliza conexiones de salto o *skip*, lo que mejora el aprovechamiento del contexto de las imágenes. Una de las ventajas adicionales de UNet es que, según sus autores, este modelo puede funcionar con relativamente pocos datos de entrenamiento, siendo esto especialmente beneficioso para tareas de segmentación de imágenes médicas y de teledetección, las cuales tienen datos anotados limitados.

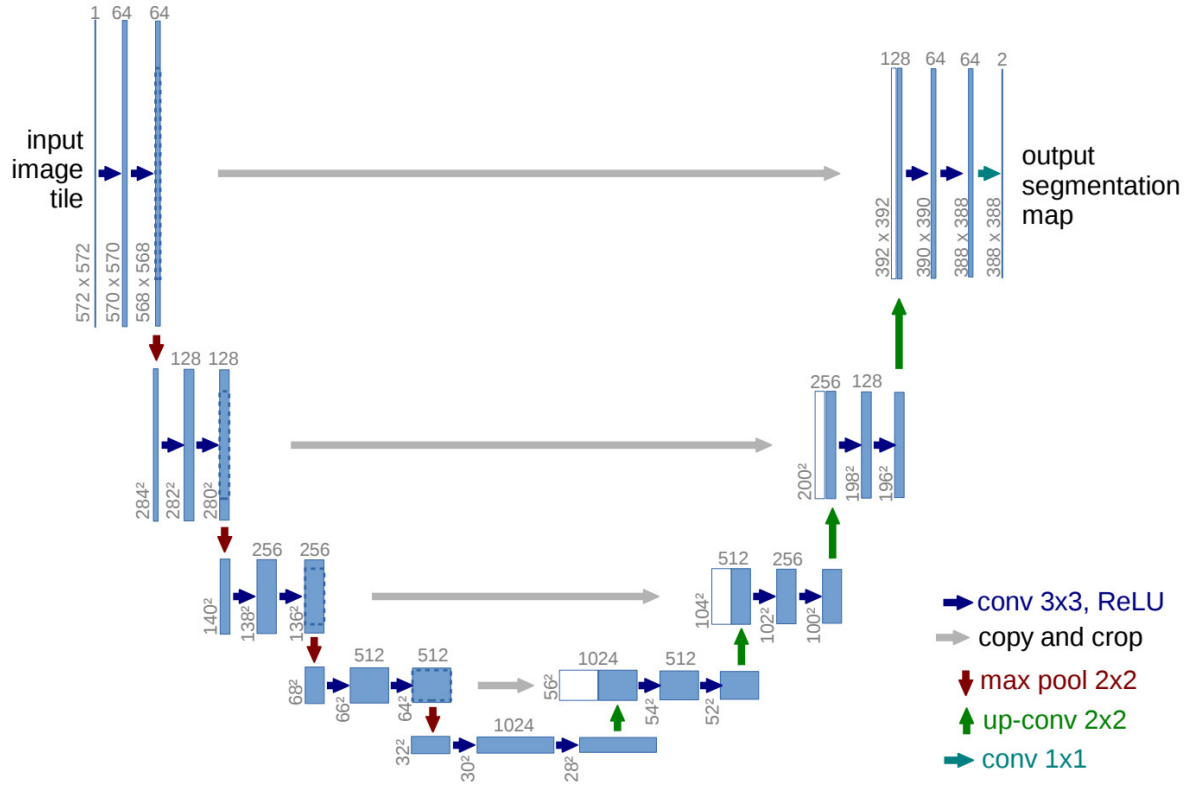
Dados los excelentes resultados de UNet se aplicó para diferentes tareas, en el campo biomédico, por ejemplo, en la segmentación de tumores cerebrales mediante imágenes de resonancia magnética (Lachinov et al., 2018). En el campo de la teledetección, para la segmentación de diversos elementos de cobertura terrestre (Ghosh et al., 2018).

Arquitectura de UNet

La arquitectura UNet es una red de tipo codificador-decodificador (*encoder-decoder*). La etapa del codificador consiste, en términos generales, en la captura de características a diferentes escalas, reduciendo las dimensiones espaciales de la imagen en cada nivel; mientras que en la etapa del decodificador se reconstruye la imagen a partir de resoluciones espaciales menores hasta cerca de su dimensión original. Uno de los aportes de UNet, es la adición de conexiones de salto o *skip*, que combinan la información del codificador en la reconstrucción, permitiendo obtener información de contexto de la imagen. La arquitectura UNet fue propuesta para la segmentación de imágenes producidas por microscopio, por lo cual, era para imágenes de un solo canal. La arquitectura original de UNet se muestra en la Figura 3.15.

Se diferencia claramente en la Figura 3.15 los 4 elementos principales de la arquitectura,

Figura 3.15
Arquitectura UNet original.



Nota: Extraído de (Ronneberger et al., 2015).

a) a la izquierda, el codificador (que consta de 4 niveles), b) a la derecha, el decodificador (también con 4 niveles), c) la conexión cuello de botella (*bottleneck*, que conecta codificador y decodificador) y d) las conexiones de salto entre los niveles.

Se observa que la entrada al modelo es una imagen de un solo canal con dimensiones 572×572 píxeles ($572 \times 572 \times 1$). En cada nivel se aplican dos convoluciones seguidas con un kernel o filtro de dimensiones 3x3, cada una seguida por la función de activación *rectified linear unit* (ReLU) que introduce no linealidades. En la implementación original de UNet, las convoluciones aplicadas se realizan sin *padding* o relleno, por lo cual, en cada aplicación de esta convolución 3×3, las dimensiones espaciales se reducen en dos píxeles tanto en alto como ancho. Por ejemplo, en el nivel más alto del codificador, las dimensiones de la imagen de entrada se reducen de 572×572 a 570×570 píxeles y así sucesivamente. Después de la aplicación de estas convoluciones, se aplica un *max pooling* con un filtro 2×2, que reduce las dimensiones de la imagen a la mitad. Además, se observa que el número de filtros en cada nivel es el doble

del anterior, comenzando con 64 filtros en el nivel más alto, pasando a 128, 256 y 512 en el último nivel. Esta etapa de codificador se conoce como camino contractivo (*contractive path*).

Seguidamente, se observa la conexión entre el codificador y decodificador. El último nivel del codificador resulta en un tensor de $64 \times 64 \times 512$, es decir, una imagen con resolución espacial de 64 píxeles de lado y de 512 canales. La conexión cuello de botella reduce una vez más estas dimensiones espaciales y aumenta el número de filtros hasta 1024 canales, siendo este mapa de características el que contiene la información de más alto nivel de los datos de entrada ($28 \times 28 \times 1024$).

La arquitectura UNet es casi simétrica, por lo que es necesario que la información del codificador y decodificador coincidan, tanto en dimensiones espaciales como en número de canales. Por ello, la salida del cuello de botella se reduce a 512 canales mediante la aplicación de una convolución *Pointwise*, coincidiendo con el número de canales del último nivel del codificador. Además de ello, se aumentan las dimensiones espaciales al doble, mediante la utilización de la operación de convolución traspuesta. En este proceso se utilizan las conexiones de salto, concatenando las informaciones capturadas en los niveles correspondientes del codificador, esta información se combina linealmente y se reducen sus canales a 512 para el nivel más profundo del decodificador. Este proceso se realiza para cada nivel del decodificador, hasta obtener un tensor con dimensiones $388 \times 388 \times 64$ que representa una imagen de 388 píxeles de lado y 64 canales. El decodificador se conoce también como camino expansivo (*expansive path*). Estos 64 canales se reducen a dos del mismo tamaño espacial que representan las máscaras finales de segmentación de salida.

Ventajas y Desventajas de UNet

Gracias a sus conexiones de salto, el modelo UNet puede obtener información de contexto, ya que combina los datos aumentados espacialmente del decodificador con los datos del codificador. Además, se ha probado que este tipo de arquitectura puede generalizar adecuadamente en tareas de segmentación incluso con conjuntos de datos pequeños (Ronneberger et al., 2015 entrenaron su modelo apenas con 30 imágenes y la aplicación de técnicas de aumento de datos).

UNet ha probado ser una excelente alternativa para la tarea de segmentación en general,

pero cuando se requiere que la precisión sea más elevada, como en tareas de detección de anormalidades o lesiones en imágenes médicas, UNet puede dar resultados no suficientemente óptimos (Z. Zhou et al., 2018). La tarea de segmentación de lagunas, por su parte, es una tarea que requiere una precisión elevada, especialmente cuando se pretende realizar estudios de monitoreo remoto. (X. Zhang et al., 2022) indican que la aplicación directa de redes FCN o UNet sin ninguna modificación en la extracción de superficies de agua producen resultados con pobre precisión y bordes borrosos.

Por otro lado, UNet puede requerir considerables recursos computacionales debido a su cantidad de parámetros, aun a pesar de no ser de las redes más pesadas ni complejas.

Implementación del Modelo Línea de Base UNet

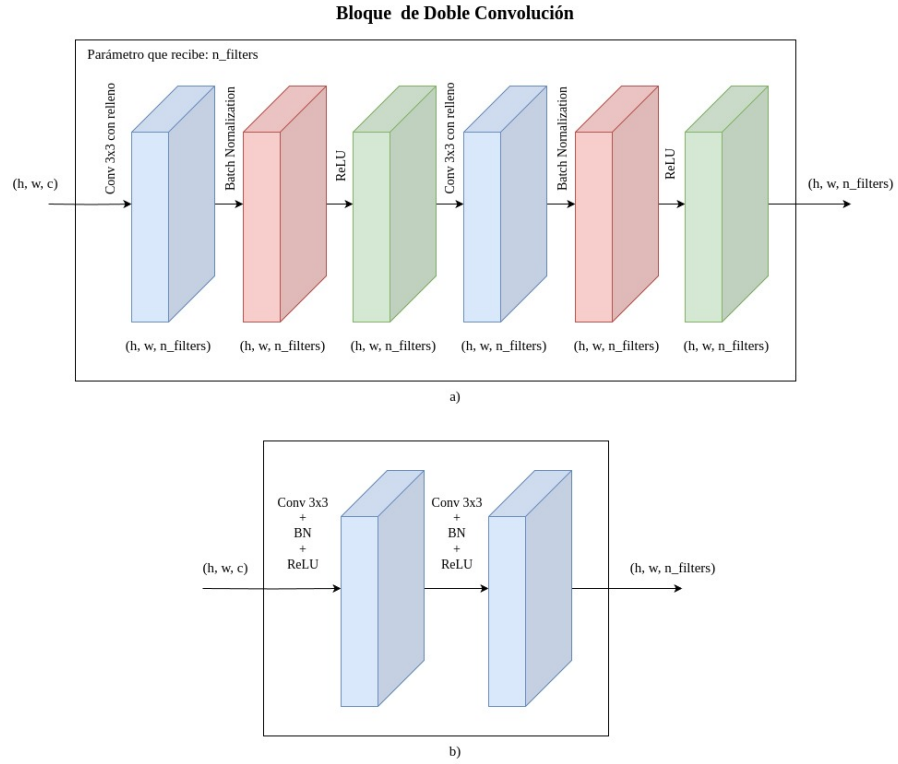
Variantes de UNet más recientes, como la propuesta de (Z. Zhang et al., 2018), aplican *zero padding* o relleno de ceros para las capas convolucionales, lo que permite mantener la resolución espacial de las imágenes y prevenir la pérdida de píxeles en los bordes, aunque con el problema de perder cierta precisión en los bordes de la imagen. Para la implementación de UNet como línea de base se toma en cuenta esta modificación, por lo que para la aplicación de las capas convolucionales, se configura el relleno de ceros para que la imagen no pierda resolución espacial. Dado que en esta investigación se está procesando imágenes satelitales con dimensiones espaciales de $256 \times 256 \times 6$, es decir, imágenes de 256×256 píxeles y de 6 canales, la entrada del modelo línea de base UNet tendrá estas dimensiones de entrada. Finalmente, la salida del modelo será de un solo canal, ya que el problema es de segmentación binaria, lo cual indica que solo hace falta un canal para describir totalmente la salida. El modelo línea de base UNet se describe completamente en las secciones codificador, cuello de botella, decodificador y conexiones de salto. El código de implementación de UNet clásico se muestra en el Apéndice B.

Codificador

La implementación de la etapa contractiva o codificador del modelo línea de base consiste en 4 niveles en los cuales las dimensiones espaciales se reducen en un factor de 2 por nivel, mientras que la cantidad de filtros se duplica por nivel.

Cada nivel del codificador se puede abstraer en un bloque **Doble convolución**, cuyo diagrama de bloques se muestra en la Figura 3.16.

Figura 3.16
Bloque de Doble convolución del codificador.



Se observa en la Figura 3.16 a) que el bloque de Doble convolución consta de 6 capas: una capa convolucional con tamaño de kernel 3×3, configurada con un relleno de ceros que permite mantener la resolución espacial de la imagen; una capa de normalización por lotes (*batch normalization*, BN); una capa de función de activación *rectified linear unit*, ReLU; otra capa convolucional; otra capa de BN y otra capa ReLU. En b) se observa este bloque simplificado, juntando tanto la capa convolucional, la capa de BN así como la capa de activación en un solo bloque.

En la Figura 3.16 se muestra que el bloque recibe como entrada un tensor con forma (h, w, c), donde h representa la altura de la imagen; w, el ancho y c, el número de canales de entrada. El bloque recibe como parámetro la variable *n_filters* que determina cuántos canales o filtros de salida tendrá el bloque. Se observa también que las dimensiones espaciales de salida del bloque son iguales a las de la entrada, esto producto de la aplicación del relleno de ceros.

Entre cada bloque de Doble convolución se aplica una capa de *max pooling* con un kernel de

tamaño 2×2 y un paso (*stride*) de 2. La operación de *max pooling*, siguiendo el funcionamiento descrito en la Sección 2.8, toma el valor máximo dentro de cada bloque o ventana de contexto de 2×2 tomando dos píxeles por paso, lo cual produce como salida una imagen con dimensiones espaciales de la mitad del valor de los datos de entrada. La aplicación de *max pooling* permite obtener información de la imagen más comprimida, obteniendo con menos datos un contexto más global de la imagen, ya que cada píxel comprime información espacial en un solo valor. Para evitar el sobreajute (*overfitting*) del modelo, se aplica una capa de *dropout* que indica el porcentaje de neuronas que se apagarán aleatoriamente durante el entrenamiento. Para esta implementación, se toma un valor común de 10 %.

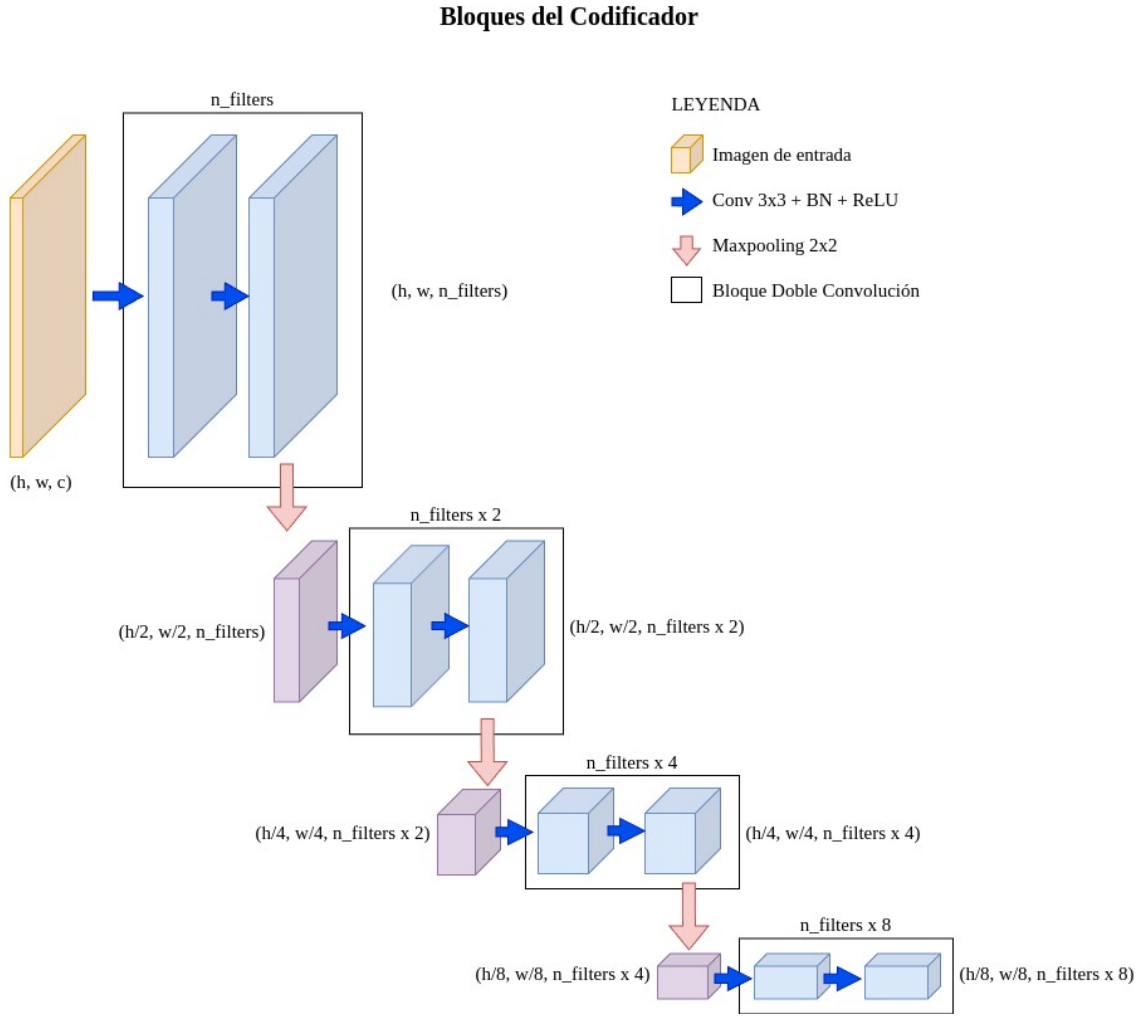
El codificador completo se observa en la Figura 3.17. Asumiendo un tensor de entrada con forma (h, w, c), las dimensiones espaciales h y w se reducen a la mitad en cada nivel, resultando en el último nivel del codificador una dimensión espacial de $(h/8, w/8)$, es decir, un octavo de las dimensiones de entrada. Por otro lado, el número de filtros o canales por cada nivel se multiplica por dos, considerando una cantidad de *n_filters* en el primer nivel, al finalizar el codificador se obtienen *n_filters*×8 filtros o canales. Entonces, sea una entrada con forma (h, w, c), el codificador devuelve un tensor con forma $(h/8, w/8,$ *n_filters*$\times 8)$.

Cuello de Botella

El cuello de botella o *bottleneck* es el bloque que conecta el codificador con el decodificador. Su función es la de reducir una vez más las dimensiones de los datos de entrada y aumentar el número de canales o filtros, generando el mapa de características más profundo de la arquitectura.

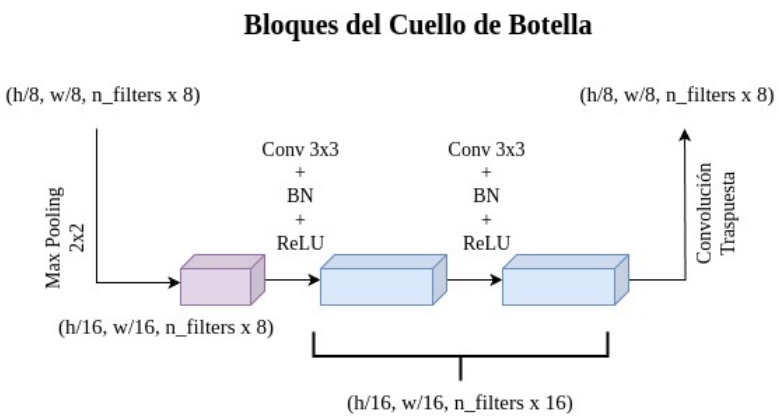
En la Figura 3.18 se observa el diagrama de bloques de la conexión cuello de botella. Se observa que la entrada a esta etapa es la salida del codificador, se aplica un *max pooling* para reducir las dimensiones espaciales a la mitad, seguidamente, se aplica el bloque de Doble convolución, obteniendo el doble de filtros que a la entrada. Para implementar la simetría de UNet, es necesario que el número de canales del nivel que sigue a continuación en el decodificador sea igual que del codificador, por ello, se aplica la capa de convolución traspuesta, que en primer lugar realiza un sobremuestreo o *upsampling* en las dimensiones espaciales, duplicando el tamaño del mapa de características, mientras que también reduce la cantidad de

Figura 3.17
Diagrama de bloques del codificador del modelo de base UNet.



filtros haciendo coincidir con el número de filtros del codificador en su mismo nivel, de tal forma que la entrada y la salida del cuello de botella tienen las mismas dimensiones y filtros.

Figura 3.18
Diagrama de bloques de cuello de botella.

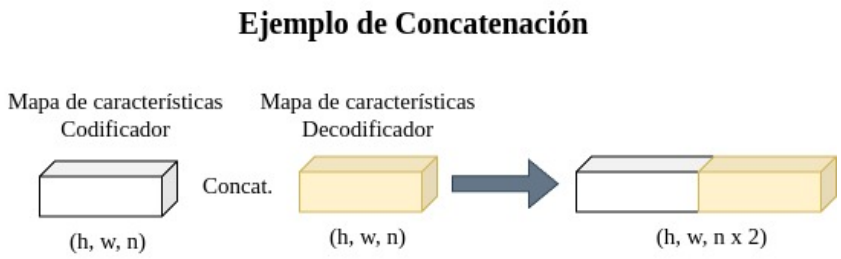


Decodificador y Conexiones de Salto

El decodificador representa el camino expansivo (*expansive path*) de la arquitectura. Se dice así porque en esta etapa se vuelve a obtener las dimensiones espaciales de la imagen original, especialmente mediante la aplicación continua de convoluciones traspuestas que son capaces de remuestrear los datos de entrada y expandirlos, en este caso, en un factor de 2. Además, aplica convoluciones con kernel de tamaño 2×2, lo que asegura un remuestreo al doble de las dimensiones espaciales.

Un punto primordial en la arquitectura de UNet es la implementación de conexiones de salto. La forma de implementar esta configuración es mediante la concatenación del mapa de características del respectivo nivel del codificador y decodificador, ya que ambos poseen las mismas dimensiones espaciales y número de canales. Esto puede apreciarse en la Figura 3.19.

Figura 3.19
Ejemplo de concatenación entre el mapa de características de codificador y decodificador.

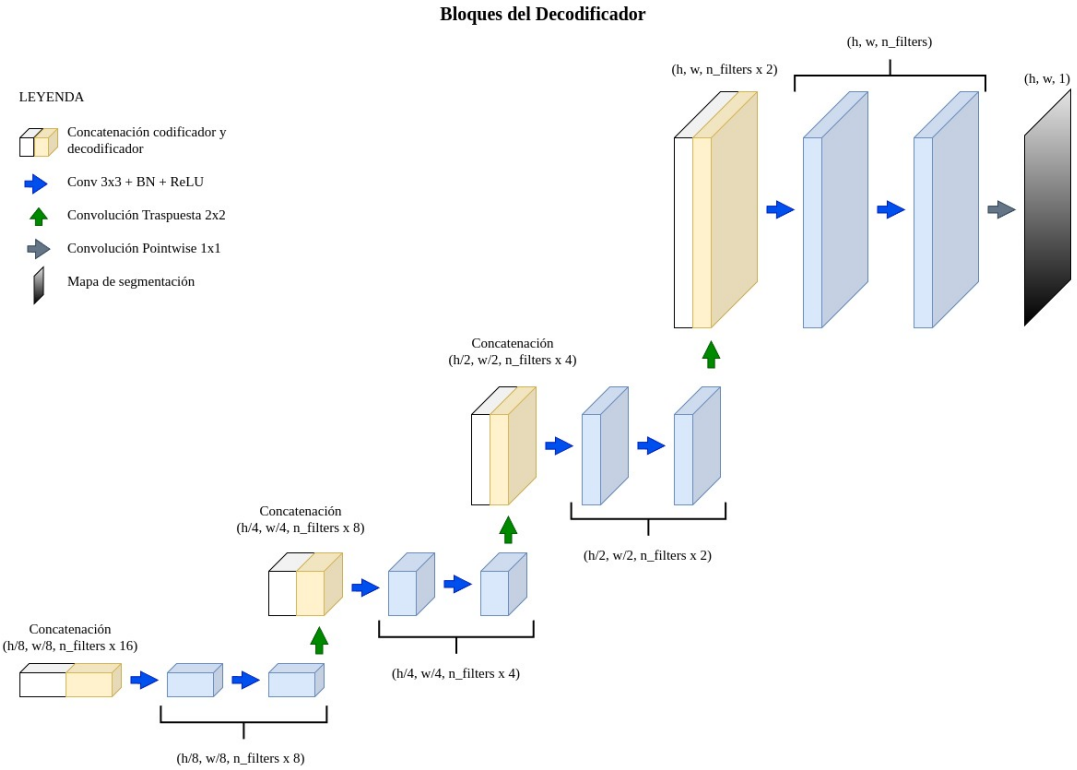


Se observa que el mapa de características generado en el decodificador se concatena o apila con el del codificador, generando un nuevo mapa de características que contiene tanto la información remuestreada o con *upsampling* como la información abstraída en el codificador, permitiendo obtener un contexto más detallado de la imagen. Este nuevo mapa de características tiene el doble de canales.

Para mantener la simetría de la arquitectura, los canales del resultado de la concatenación deben reducirse a la mitad, para ello, se vuelve a aplicar el bloque de Doble convolución, manteniendo las dimensiones espaciales y reduciendo el número de canales. En la Figura 3.20 se aprecia la estructura y bloques del decodificador combinando con el proceso de concatenación de mapa de características.

Se observa en la Figura 3.20 que la entrada a esta etapa es la concatenación de mapa de

Figura 3.20
Diagrama de bloques del decodificador del modelo línea de base UNet.



características del codificador y decodificador, estos pasan a través del bloque de Doble convolución, reduciendo el número de canales a la mitad. Seguidamente, se aplica una convolución traspuesta que remuestrea las dimensiones espaciales del mapa de características al doble, subiendo al siguiente nivel y concatenando con el respectivo mapa del codificador. Estos pasos se realizan sucesivamente hasta que se recupera la dimensión original. En este último nivel se obtienen *n_filters*, los cuales se reducen a un solo canal de salida con forma $(h, w, 1)$ (mapa de segmentación) mediante la aplicación de una convolución punto a punto o *Pointwise* y una función de activación Sigmoide, este último canal representa la probabilidad de que un determinado píxel sea clasificado o no como laguna, siendo los valores cercanos a 1, los píxeles con más alta probabilidad, esto es representado en la Figura 3.20 como un gradiente de color negro y blanco.

Arquitectura del Modelo Línea de Base UNet Implementado

La arquitectura del modelo se puede resumir en los siguientes puntos:

- Aplicación de bloques de Doble convolución con kernel 3×3, manteniendo la resolución

espacial y aumentando/reduciendo el número de filtros o canales, junto con capas de normalización por lotes y función de activación ReLU.

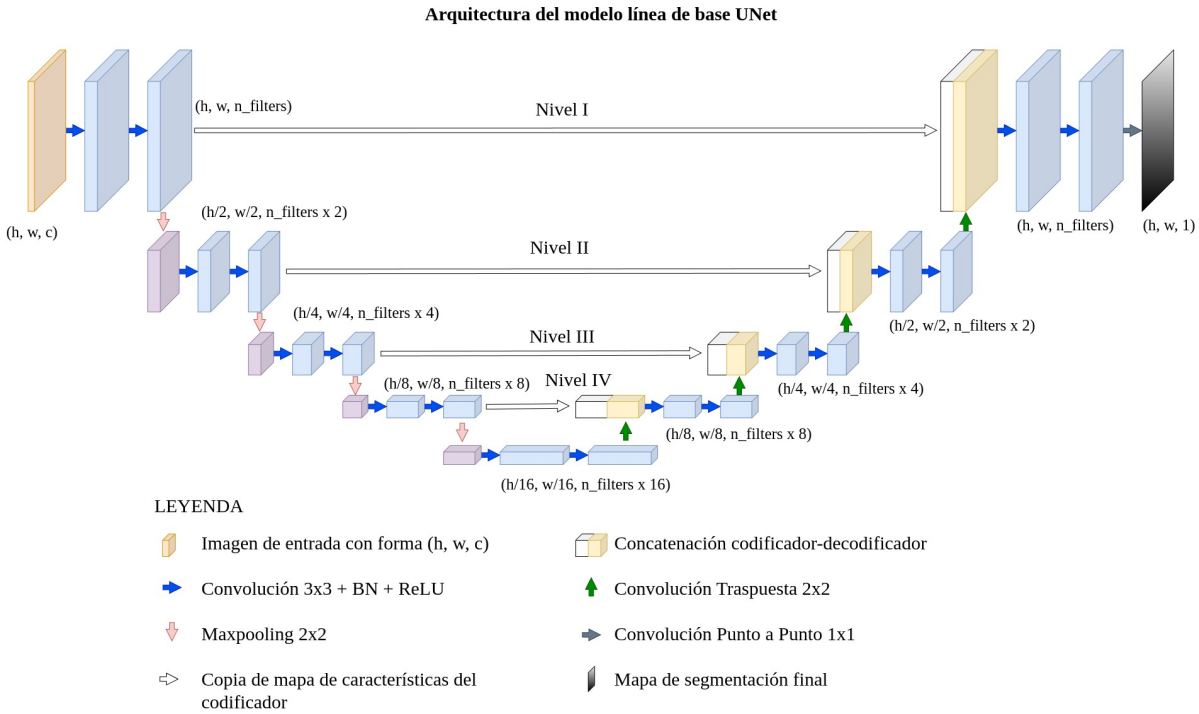
- Aplicación de capa *max pooling* con filtro 2×2 para reducir las dimensiones espaciales de los datos en cada nivel del codificador.

- Aplicación de convolución traspuesta con filtro 2×2 para remuestrear o aumentar las dimensiones espaciales de los datos de entrada al doble, reduciendo también el número de filtros o canales a la mitad en cada nivel del decodificador.

- Aplicación de convolución punto a punto o *Pointwise* con filtro 1×1 para reducir los *n_filters* a un solo canal de predicción de probabilidad.

El diagrama de bloques del modelo línea de base UNet se muestra en la Figura 3.21, que resume los puntos anteriores, incluyendo las formas de los tensores de cada nivel.

Figura 3.21
Arquitectura completa del modelo línea de base UNet.



Se observa en esta representación la composición casi simétrica del modelo, con su característica forma de U, que da nombre al modelo. La implementación de esta arquitectura permite como entrada a imágenes de cualesquiera dimensiones espaciales (alto, ancho) y número de

canales (c) descritos por (h, w, c). El parámetro que define la profundidad del modelo es el número de filtros o *n_filters*, los cuales se van duplicando a cada nivel. La arquitectura UNet original implementó 64 filtros en el primer nivel, llegando hasta 1024 ($64 \times 16 = 1024$) filtros para el mapa de características del cuello de botella que contiene a información de más alto nivel abstraída de la imagen de entrada. En la Tabla 3.12 se resumen las formas de entrada y salida de cada nivel tanto del codificador como del decodificador.

Tabla 3.12
Formas de entrada y salida de cada nivel del modelo UNet.

	Codificador		**Decodificador**	
Niveles	**Entrada**	**Salida**	**Entrada**	**Salida**
Nivel I	(h, w, c)	(h, w, n_f)	$(h, w, n_f \times 2)$	$(h, w, 1)$
Nivel II	(h, w, n_f)	$(\frac{h}{2}, \frac{w}{2}, n_f \times 2)$	$(\frac{h}{2}, \frac{w}{2}, n_f \times 4)$	$(\frac{h}{2}, \frac{w}{2}, n_f \times 2)$
Nivel III	$(\frac{h}{2}, \frac{w}{2}, n_f \times 2)$	$(\frac{h}{4}, \frac{w}{4}, n_f \times 4)$	$(\frac{h}{4}, \frac{w}{4}, n_f \times 8)$	$(\frac{h}{4}, \frac{w}{4}, n_f \times 4)$
Nivel IV	$(\frac{h}{4}, \frac{w}{4}, n_f \times 4)$	$(\frac{h}{8}, \frac{w}{8}, n_f \times 8)$	$(\frac{h}{8}, \frac{w}{8}, n_f \times 16)$	$(\frac{h}{8}, \frac{w}{8}, n_f \times 8)$

Donde n_f representa el parámetro $n\_filters$ que se va duplicando en cada nivel del modelo.

3.2.2. Representación de una Imagen en el Dominio de la Frecuencia

Dado que para la presente investigación se hará uso de las transformaciones de las imágenes en la frecuencia , en esta sección se define formalmente la representación matemática de una imagen, sus transformaciones en la frecuencia así como la aplicación de los filtros.

(Oppenheim et al., 1998) indican que cualquier fenómeno físico puede ser descrito por una señal que captura su información a través de las variaciones inherentes a la naturaleza del fenómeno. Estas señales pueden ser representadas matemáticamente mediante funciones de una o más variables independientes.

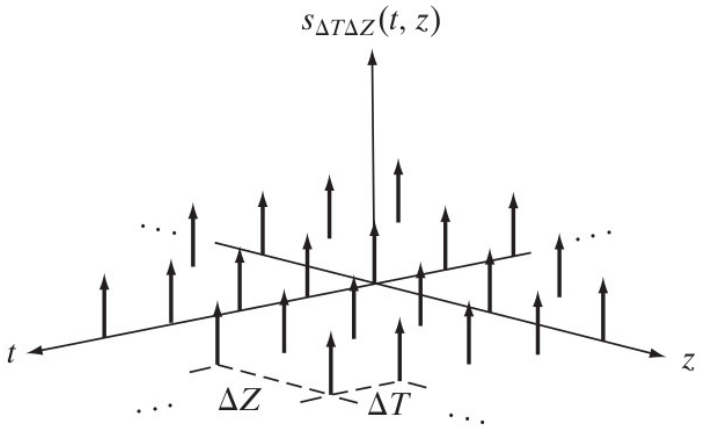
Una imagen es la proyección en dos dimensiones del espacio tridimensional, representa la intensidad o brillantez de la variable que se esté midiendo. A diferencia de otro tipo de señales cuya variable independiente es el tiempo (señales temporales como la tensión o corriente), las imágenes son señales de tipo espacial, cuyas variables independientes son las coordenadas espaciales. Idealmente una imagen puede ser descrita como una función $f(t, z)$, donde t y z representan las 2 variables espaciales (2D), continuas e independientes de la imagen. El valor de $f(t, z)$ representa el nivel de brillantez o intensidad de la imagen en (t, z).

Sin embargo, en la vida real no puede obtenerse una imagen con variables espaciales continuas sino discretas, producto de las limitaciones al momento de medir las variaciones de intensidad con los sensores. Entonces, esta discretización se logra mediante el muestreo de la señal continua, lo cual, se consigue matemáticamente multiplicando $f(t, z)$ por la función de muestreo (*sampling*) que es un tren de impulsos en dos dimensiones, la cual se define como en la Ecuación 3.8.

$$s_{\Delta T \Delta Z}(t, z) = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} \delta(t - x\Delta T, z - y\Delta Z) \tag{3.8}$$

$s_{\Delta T \Delta Z}(t, z)$ es una función de variables continuas (t, z), que representa impulsos igualmente espaciados en ΔT y ΔZ, con x y y siendo enteros, cuya representación gráfica se muestra en la Figura 3.22.

Figura 3.22
Tren de impulsos en dos dimensiones.



Nota: Extraído de (Gonzales y Woods, 2008).

Se realiza la multiplicación del tren de impulsos con la función $f(t, z)$ y se obtiene la función muestreada como se aprecia en la Ecuación 3.9.

$$\tilde{f}(t, z) = f(t, z)s_{\Delta T \Delta Z}(t, z) = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} f(t, z)\delta(t - x\Delta T, z - y\Delta Z) \tag{3.9}$$

Donde $\tilde{f}(t, z)$ es una función de variables continuas y representa a $f(t, z)$ muestreada a una tasa de ΔT y ΔZ en cada dimensión espacial. Para determinar el valor de cada punto muestreado, se integra la función, siguiendo la Ecuación 3.10.

$$f_{x,y} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t,z)\delta(t - x\Delta T, z - y\Delta Z)dtdz = f(x\Delta T, y\Delta Z) \tag{3.10}$$

Aprovechando la propiedad de filtrado (*sifting property*) de la función Delta de Dirac, se puede obtener el valor de la función evaluada en los puntos $x\Delta T$ y $y\Delta Z$. La agrupación de todos los valores de $f_{x,y}$ forma una función de variables discretas $f(x, y)$ para $-\infty < x < \infty$ y $-\infty < y < \infty$, con x e $y \in \mathbb{Z}$.

Se observa que, idealmente, esta imagen muestreada se extiende desde $-\infty$ hasta ∞ para sus dos variables, sin embargo, en la práctica, una imagen digitalizada tiene dimensiones espaciales restringidas, por ejemplo, de $M \times N$ de tamaño.

Transformada Discreta de Fourier en 2D y su Inversa

Siguiendo las Ecuaciones 2.21 y 2.22 de la Sección 2.10, en este caso aplicada para variables de dos dimensiones en la que se toman $M \times N$ muestras equidistantes de un periodo de la transformada de Fourier de la señal muestreada, la cual es infinita, continua y periódica, se obtiene la transformada de Fourier discreta (DFT), la cual se define como se muestra en la Ecuación 3.11.

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)} \quad u = 0, 1, 2, \ldots, M-1 \quad v = 0, 1, 2, \ldots, N-1 \tag{3.11}$$

Los valores de M y N se basan en las dimensiones de la imagen digital $f(x, y)$. Se considera a u y v como las variables discretas de la frecuencia en el espacio, pudiendo representarse $F(u, v)$ como una matriz de también $M \times N$ elementos. $F(u, v)$ es una función compleja, por lo que esta puede ser representada en términos de su parte real e imaginaria, siendo ambas funciones de u y v, así como en forma polar, tal como se observa en la Ecuación 3.12.

$$F(u,v) = R(u,v) + jI(u,v) = |F(u,v)|e^{j\phi(u,v)} \tag{3.12}$$

Existen 3 representaciones que se pueden obtener de la expresión 3.12: la magnitud o

espectro de frecuencia, la fase y el espectro de potencia.

$$|F(u,v)| = \left[R^2(u,v) + I^2(u,v)\right]^{1/2} \tag{3.13}$$

$$\phi(u,v) = \arctan\left[\frac{I(u,v)}{R(u,v)}\right] \tag{3.14}$$

$$P(u,v) = R^2(u,v) + I^2(u,v) \tag{3.15}$$

De las Ecuaciones 3.12, 3.13, 3.14 y 3.15, tanto $F(u,v), R(u,v), I(u,v), |F(u,v)|, \phi(u,v)$ y $P(u,v)$ son matrices de tamaño $M \times N$, por lo cual, pueden ser representadas en forma de imágenes. Generalmente, el valor de $|F|$ cuando $(u,v) = (0,0)$ es bastante mayor a los de otras frecuencias, por lo que, para representación visual, se aplica la transformación logarítmica al espectro de frecuencia mediante $(1 + \log|F(u,v)|)$.

De la misma manera, teniendo los valores discretos de la transformada de Fourier, es posible obtener $f(x,y)$ usando la transformada de Fourier discreta inversa (IDFT) como se muestra en la Ecuación 3.16.

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)} \quad x = 0,1,2,\ldots,M-1 \quad y = 0,1,2,\ldots,N-1 \tag{3.16}$$

Siendo 3.11 y 3.16 el par de transformadas de Fourier discretas.

Implementación de DFT, IDFT y la Transformada Rápida de Fourier

De acuerdo a las Ecuaciones 3.11 y 3.16, se observa que para el cálculo de sus respectivos valores solo hacen falta sumas simples, multiplicaciones y sumatorias. La implementación directa de estas ecuaciones aumentan en número de operaciones rápidamente a medida que las dimensiones de la imagen procesada crece, ya que la DFT requiere alrededor $(MN)^2$ operaciones (Gonzales y Woods, 2008).

La aplicación directa de la DFT como la IDFT en general es muy ineficiente, por lo que

el descubrimiento de la transformada rápida de Fourier (FFT) fue muy importante. La FFT es un algoritmo que aprovecha las propiedades de simetría de los exponenciales complejos (que representan senos y cosenos que son periódicos y simétricos), separando el problema de la DFT principal de forma sucesiva en dos partes computacionalmente menos complejas. Las primeras implementaciones de FFT requerían que los datos de entrada tengan 2^n elementos para ser computables, sin embargo, nuevas implementaciones ya no tienen esta limitación. La ventaja de la FFT sobre DFT es que requiere menos cálculos, siendo del orden de $MN\log(MN)$, ventaja que se aprecia más en el campo del procesamiento de imágenes a medida que los datos tienen dimensiones espaciales mayores.

Tanto la DFT como la FFT dan resultados matemáticamente idénticos, la única diferencia es el algoritmo de implementación siendo la FFT mucho más eficiente, por lo que para aplicaciones prácticas, como la presente investigación, se toma esta herramienta.

Los algoritmos de la FFT así como su inversa, la IFFT, son implementados en diversos paquetes de programación tales como Numpy, Tensorflow, Pytorch, entre otros.

Aplicación de Filtros en el Dominio de la Frecuencia

La aplicación de filtros en el dominio de la frecuencia se da simplemente por la multiplicación punto a punto. Sea una imagen de tamaño $M \times N$ definida por $f(x, y)$ siendo x e y variables espaciales discretas e independientes, cuya transformada de Fourier se define como una función $F(u, v)$ con u y v variables discretas de frecuencia, que puede ser representada como una imagen de $M \times N$ de tamaño y sea $H(u, v)$ una función filtro o la función de transferencia del filtro; en el dominio de la frecuencia se filtra la imagen de entrada mediante la multiplicación punto a punto con el filtro $H(u, v)$, como se muestra en la Ecuación 3.17.

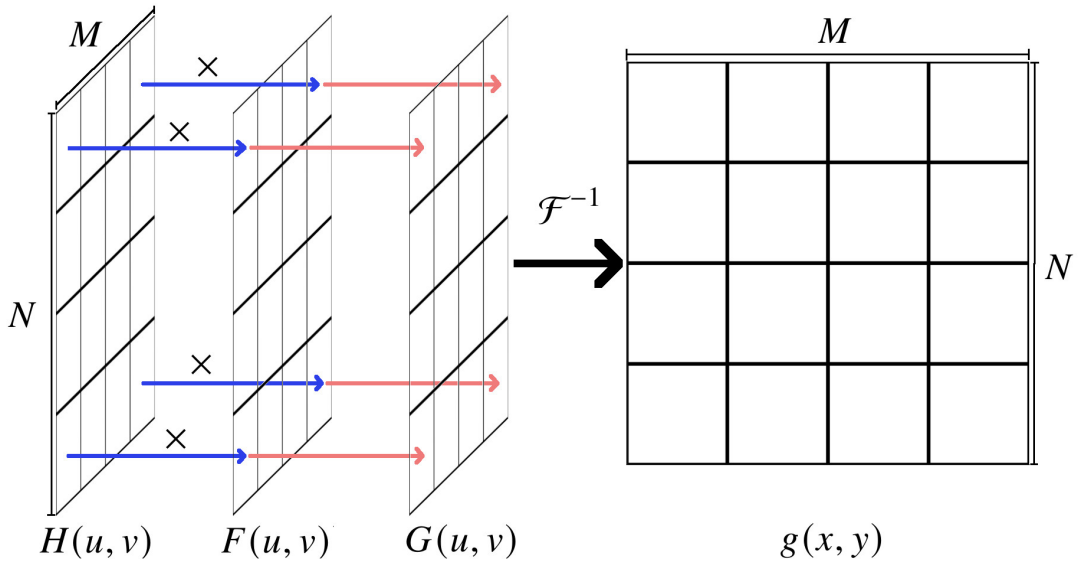
$$G(u, v) = H(u, v) \otimes F(u, v) \tag{3.17}$$

Donde $G(u, v)$ representa la imagen filtrada en el dominio de la frecuencia. Se puede determinar la imagen en el dominio espacial mediante la aplicación de la transformada de Fourier inversa, como en la Ecuación 3.18.

$$g(x, y) = \mathcal{F}^{-1}[H(u, v) \otimes F(u, v)] \tag{3.18}$$

En el dominio de la frecuencia, la aplicación del filtro es la modulación de los diferentes componentes de frecuencia de la imagen ponderados por los valores de $H(u, v)$, por lo que este último debe tener también las mismas dimensiones de $F(u, v)$, es decir, ser de $M \times N$ de tamaño. Se debe considerar que si los valores de la matriz $H(u, v)$ son reales y los valores de $f(x, y)$ también, entonces, en teoría $g(x, y)$ tiene solo valores reales. Asumiendo que $F(u, v)$ y $H(u, v)$ son matrices de tamaño $M \times N$, la Figura 3.23 representa el proceso de filtrado en el dominio de la frecuencia mediante la multiplicación punto a punto, píxel a píxel o *pixelwise*.

Figura 3.23
Filtrado en la frecuencia mediante multiplicación punto a punto.



En términos matemáticos, la multiplicación punto a punto se puede expresar de acuerdo a la Ecuación 3.19.

$$G(i, j) = H(i, j) \times F(i, j) \quad i = 0, 1, \ldots, M - 1 \quad j = 0, 1, \ldots, N - 1 \tag{3.19}$$

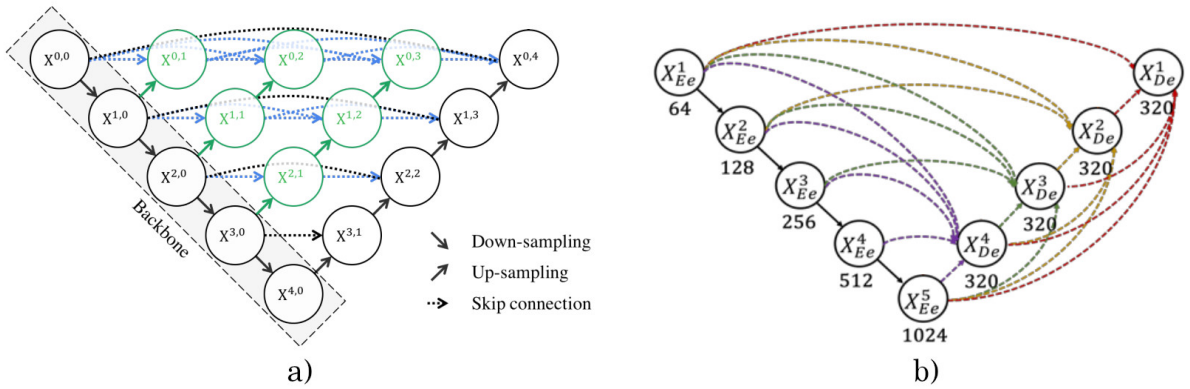
Que representa el valor de G en el punto (i, j) evaluado como el producto del valor de H multiplicado por F en el píxel (i, j), respectivamente.

3.2.3. Propuesta de Modelo Basado en UNet: UNetFFT

Para la presente investigación se propone una modificación del modelo UNet que aprovecha sus ventajas, como su capacidad de aprendizaje incluso con pocas muestras anotadas y su capacidad de utilizar información de contexto mediante las conexiones de salto; e intenta abordar la problemática de detección de detalles finos y precisos, característica importante para la tarea de segmentación de lagunas. La modificación al modelo línea de base UNet se centra específicamente en las conexiones de salto, tema abordado en diferentes investigaciones como la de (Z. Zhou et al., 2018), que propone UNet++, y la de (Huang et al., 2020), proponiendo UNet3+, ambos intentando aprovechar la información de diferentes niveles de abstracción mediante la aplicación de convoluciones densas y conexiones de salto a escala completa, respectivamente. En la Figura 3.24 se aprecia cómo se modifican las conexiones de salto de UNet en busca de aprovechar la información de niveles más profundos.

Figura 3.24

Arquitecturas de UNet con modificaciones en las conexiones de salto.



Nota: a) UNet++, aplicando convoluciones densas. b) UNet3+, aplicando convoluciones a escala completa. Extraído de (Z. Zhou et al., 2018) y (Huang et al., 2020).

Recientes investigaciones han abordado la utilización de la representación en el dominio de la frecuencia de la información procesada aprovechando que en este dominio se puede obtener información global de los datos. Se utiliza la transformada rápida de Fourier como la herramienta básica para cambiar del dominio espacial al frecuencial. La aplicación de este concepto se ha dado principalmente en el desarrollo de módulos transformadores (*Transformers*), como en (Labbihi et al., 2024) que combinaron redes CNN con transformadores en frecuencia (*frequency transformer*). Otro es el caso de (Wu et al., 2024), que implementaron un módulo llamado

spectrum-space transformer, que permite combinar dos fuentes de información diferente: una representa imágenes de ruido y la otra, características abstraídas de una imagen de entrada, esto lo logran transformando ambas entradas al dominio de la frecuencia, aplicando conceptos de *vision transformers*, así como filtros aprendibles en el dominio de la frecuencia. La desventaja principal de los módulos transformadores, es que a pesar de que pueden obtener excelentes resultados, requieren de extensas bases de datos para su entrenamiento así como de grandes capacidades computacionales debido a su gran cantidad de parámetros aprendibles.

Tomando como referencia estas dos ideas, esto es, a) la modificación de las conexiones de salto del modelo UNet y b) la aplicación de la transformada de Fourier y filtros en el dominio de la frecuencia; se pretende mejorar el rendimiento del modelo línea de base, ya que este tiene limitaciones en la detección fina de bordes y de pequeños cuerpos, debido principalmente que utiliza convoluciones con kernel 3×3, que es una ventana de contexto pequeña. Se propone en esta investigación a UNetFFT, una modificación de UNet que utiliza las conexiones de salto, que combinan información contextual del codificador y decodificador, así como la transformada de Fourier, que permite obtener información con un contexto global y la aplicación de filtros aprendibles en el dominio de la frecuencia que puedan obtener detalles más finos en las imágenes procesadas para conseguir segmentaciones más precisas.

Modificación de Conexiones de Salto y Aplicación de FFT

En el modelo línea de base UNet, las conexiones de salto consisten simplemente en la concatenación o apilamiento del mapa de características del codificador y decodificador, como se aprecia en la Figura 3.19. Luego, este conjunto de características se reducen mediante la aplicación de dos convoluciones seguidas. Para el modelo propuesto UNetFFT, se mantiene la estructura tanto del codificador, decodificador y cuello de botella, modificándose solamente las conexiones de salto, combinando la información de cada nivel del codificador y decodificador en el dominio de la frecuencia en lugar de una simple concatenación y procesamiento en el dominio espacial, para lo cual se crea un módulo que realiza esta combinación en frecuencia llamado *Fourier Combination*.

Módulo FFT: *Fourier Combination*

El módulo *Fourier Combination* tiene la función de combinar la información de codificador y decodificador en el dominio de la frecuencia. Para ello, toma los mapas de características de entrada, los pasa al dominio de la frecuencia, los procesa mediante multiplicación punto a punto con una matriz de filtros, combina linealmente tanto la información filtrada como la original (en el dominio de la frecuencia) y se aplica la transformada inversa de Fourier para volver al dominio espacial y continuar con el procedimiento de UNet en el decodificador.

El módulo recibe como entradas los mapas de características del codificador y decodificador del mismo nivel, es decir, mapas que tienen tanto las mismas dimensiones espaciales como número de canales, de forma general, C_i y D_i representan el mapa de características el nivel i del codificador y decodificador, respectivamente. Ambos siendo tensores con forma (h, w, c). A su vez, el módulo asocia dos tensores con parámetros aprendibles y de valores reales que representan los filtros en el dominio de la frecuencia, $\mathcal{M}_{Ci}$ y $\mathcal{M}_{Di}$ para el mapa de características del codificador y decodificador, respectivamente. Estos tensores deben tener la misma forma que los mapas de características, esto es, (h, w, c).

El funcionamiento del módulo *Fourier Combination* descrito de forma matemática, puede expresarse como a continuación se desarrolla. La expresión 3.20 representa los mapas de características que son entradas al módulo, estos mapas están en el dominio del espacio y representan imágenes de tamaño $h \times w$ y c canales, donde i representa el nivel del modelo.

$$C_i \quad \text{y} \quad D_i \quad \in \quad \mathbb{R}^{h \times w \times c} \qquad i = 1, 2, 3, 4 \tag{3.20}$$

Estos mapas de características son pasados al dominio de la frecuencia mediante la transformada rápida de Fourier de acuerdo a la expresión 3.21, cuyo resultado es un tensor de valores complejos con forma (h, w, c).

$$\mathcal{C}_i = \mathcal{F}[C_i] \quad \text{y} \quad \mathcal{D}_i = \mathcal{F}[D_i] \quad \in \quad \mathbb{C}^{h \times w \times c} \tag{3.21}$$

Siguiendo la metodología de (Feng et al., 2024) que trabajó también con procesamiento de imágenes en la frecuencia, se toma solamente la parte real de la transformada de Fourier, como

muestra la expresión 3.22, resultando en tensores de valores reales.

$$\mathcal{R}\{C_i\} \quad \text{y} \quad \mathcal{R}\{\mathcal{D}_i\} \quad \in \quad \mathbb{R}^{h\times w\times c} \tag{3.22}$$

Por otro lado, se tienen dos tensores de valores reales definidos en la expresión 3.23, que son tensores con iguales dimensiones a la transformada de Fourier de los mapas de características de entrada y con c filtros correspondientes a cada característica de entrada. Estos tensores son el conjunto de filtros con parámetros aprendibles, una para el codificador y otra para el codificador.

$$\mathcal{M}_{Ci} \quad \text{y} \quad \mathcal{M}_{Di} \quad \in \quad \mathbb{R}^{h\times w\times c} \tag{3.23}$$

Teniendo como presupuestos a la parte real de la FFT del codificador y decodificador, así como los filtros para cada uno de ellos, se realiza la multiplicación punto a punto de la FFT y su filtro correspondiente, como se aprecia en las expresiones 3.24 y 3.25, donde $\otimes$ representa la multiplicación punto a punto, definida en la Ecuación 3.19.

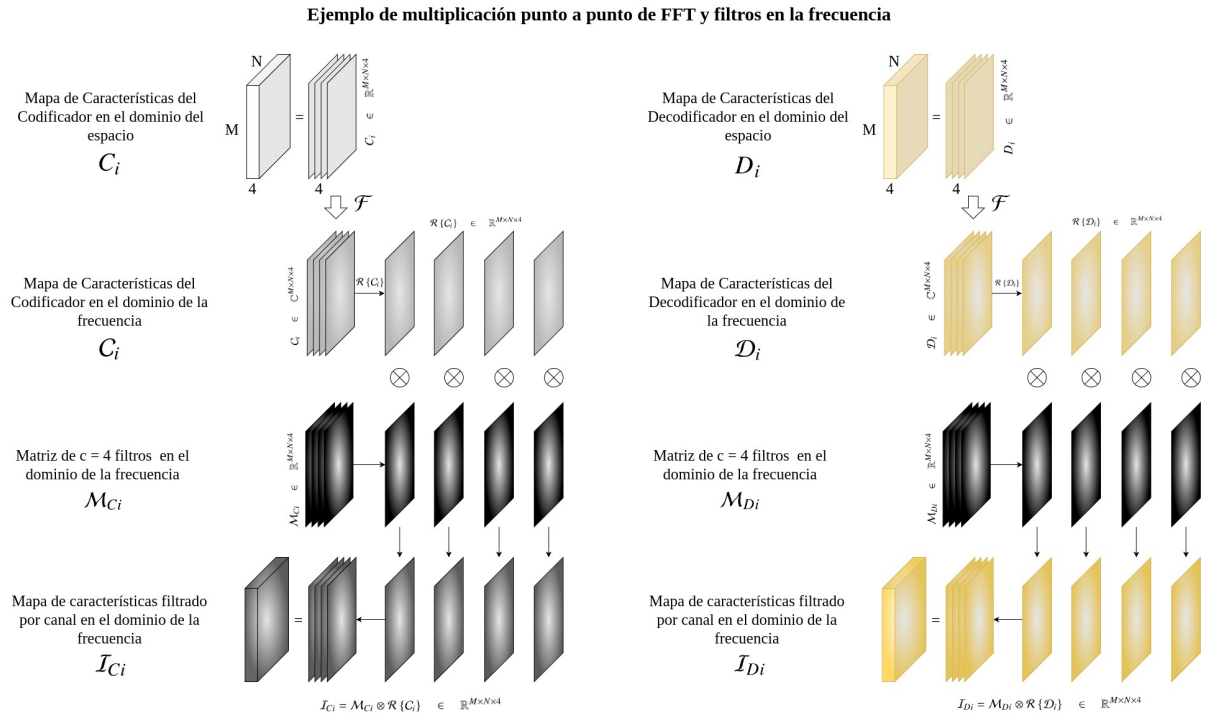
$$\mathcal{I}_{Ci} = \mathcal{M}_{Ci} \otimes \mathcal{R}\{C_i\} \quad \in \quad \mathbb{R}^{h\times w\times c} \tag{3.24}$$

$$\mathcal{I}_{Di} = \mathcal{M}_{Di} \otimes \mathcal{R}\{\mathcal{D}_i\} \quad \in \quad \mathbb{R}^{h\times w\times c} \tag{3.25}$$

Aunque no es tan intuitivo, el producto punto a punto directo de los tensores con forma (h, w, c) equivale a realizar el producto punto a punto de cada canal con su propio filtro (multiplicación punto a punto de matrices). Esto puede apreciarse mejor en la Figura 3.25, que representa gráficamente un ejemplo de cómo se computa este proceso hasta la multiplicación de las matrices, para un mapa de características con dimensiones espaciales $M \times N$ y de 4 canales.

En la Figura 3.25 se observa de forma gráfica cómo es el proceso de filtrado en el dominio de la frecuencia, realizándose mediante la multiplicación punto a punto de cada canal del mapa de características de entrada con su respectivo filtro de parámetros aprendibles, obteniéndose, hasta estos pasos, el mapa de características filtrado en el dominio de la frecuencia tanto del codificador como decodificador, denotados con $\mathcal{I}_{Ci}$ y $\mathcal{I}_{Di}$, respectivamente.

Figura 3.25
Ejemplo de multiplicación punto a punto de mapa de características.



Con los resultados de filtrado del codificador y decodificador, se procede a la combinación lineal tanto de la parte real de la FFT del mapa de características sin procesar ($\mathcal{R}\{C_i\}$ y $\mathcal{R}\{\mathcal{D}_i\}$) como del mapa de características filtrado ($\mathcal{I}_{Ci}$ y $\mathcal{I}_{Di}$). Para ello, primero se realiza la concatenación de la información en frecuencia, como se muestra en la expresión 3.26. La utilización de los datos sin procesar se debe a que esto permite al modelo en su fase de entrenamiento evitar el problema de desvanecimiento del gradiente al actualizar los pesos después de cada iteración.

$$Conc = Concat(\mathcal{I}_{Ci}, \mathcal{R}\{C_i\}, \mathcal{I}_{Di}, \mathcal{R}\{\mathcal{D}_i\}) \quad \in \quad \mathbb{R}^{h\times w\times(4\times c)} \tag{3.26}$$

Donde, debido a la concatenación, el número de canales de la variable $Conc$ se multiplica por 4. Para realizar la combinación lineal de los datos se aplica una convolución *Pixelwise* o punto a punto, reduciendo el número de canales de $(4\times c)$ a c. Luego, se utiliza una función de activación no lineal ReLU, como se muestra en la expresión 3.27.

$$Combination_f = ReLU(ConvPixelwise(Conc)) \quad \in \quad \mathbb{R}^{h\times w\times c} \tag{3.27}$$

Donde $Combination_f$, representa la combinación lineal de los datos procesados en el dominio de la frecuencia. Como paso final y último procedimiento del módulo, se aplica la FFT Inversa (IFFT) para volver al dominio espacial y seguir procesando los datos con el decodificador del modelo, como se muestra en la expresión 3.28. A pesar de que los valores de $Combination_f$ son reales, nada asegura que su transformada de Fourier inversa dé valores reales, debido a que se deben cumplir ciertos criterios de simetría, además que en el cálculo computacional puede haber ciertos errores numéricos y de aproximación que producen valores complejos parásitos (Gonzales y Woods, 2008), por lo que para abordar esta circunstancia, se toma el valor absoluto de la IFFT.

$$Combination_e = \left\| \mathcal{F}^{-1} \left\{ Combination_f \right\} \right\| \quad \in \quad \mathbb{R}^{h \times w \times c} \tag{3.28}$$

Donde $Combination_e$ es el resultado del proceso de filtrado y combinado en la frecuencia, pero convertido al dominio del espacio.

El diagrama de bloques del módulo que resume este comportamiento se muestra en la Figura 3.26. El módulo *Fourier Combination* tiene como entrada dos mapas de características en el dominio del espacio (del codificador y decodificador) cada uno con forma (h, w, c), los procesa en el dominio de la frecuencia, y devuelve un mapa de características final en el dominio del espacio con forma (h, w, c).

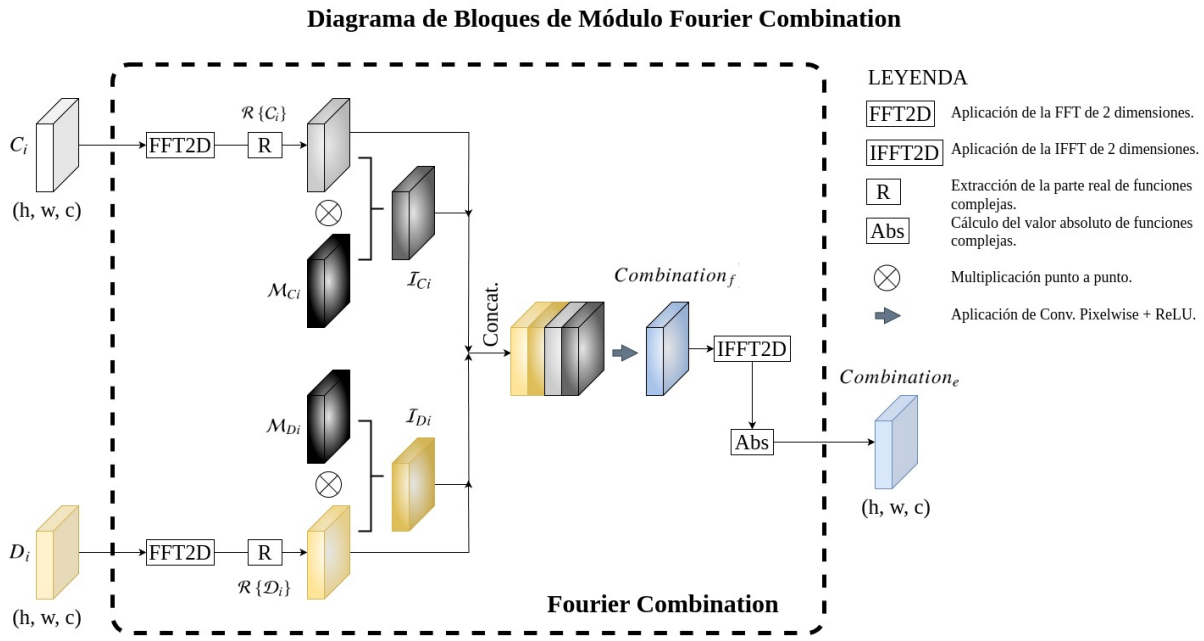
Arquitectura de UNetFFT

Se añade el módulo *Fourier Combination* en cada uno de los 4 niveles del modelo UNet, específicamente en las conexiones de salto, como se muestra en la Figura 3.27. Se observa de forma más evidente en esta representación cómo el módulo en la frecuencia utiliza como presupuestos a los mapas de características del codificador y decodificador de su respectivo nivel, mientras que el resultado se pasa al decodificador para continuar con el procesamiento en el dominio del espacio.

Para la implementación e incrustación del módulo, se crean 4 instancias de *Fourier Combination*, siempre tomando en cuenta las dimensiones espaciales y número de canales de cada nivel para crear y asociar correctamente el conjunto de matrices filtro aprendibles. Es evidente

Figura 3.26

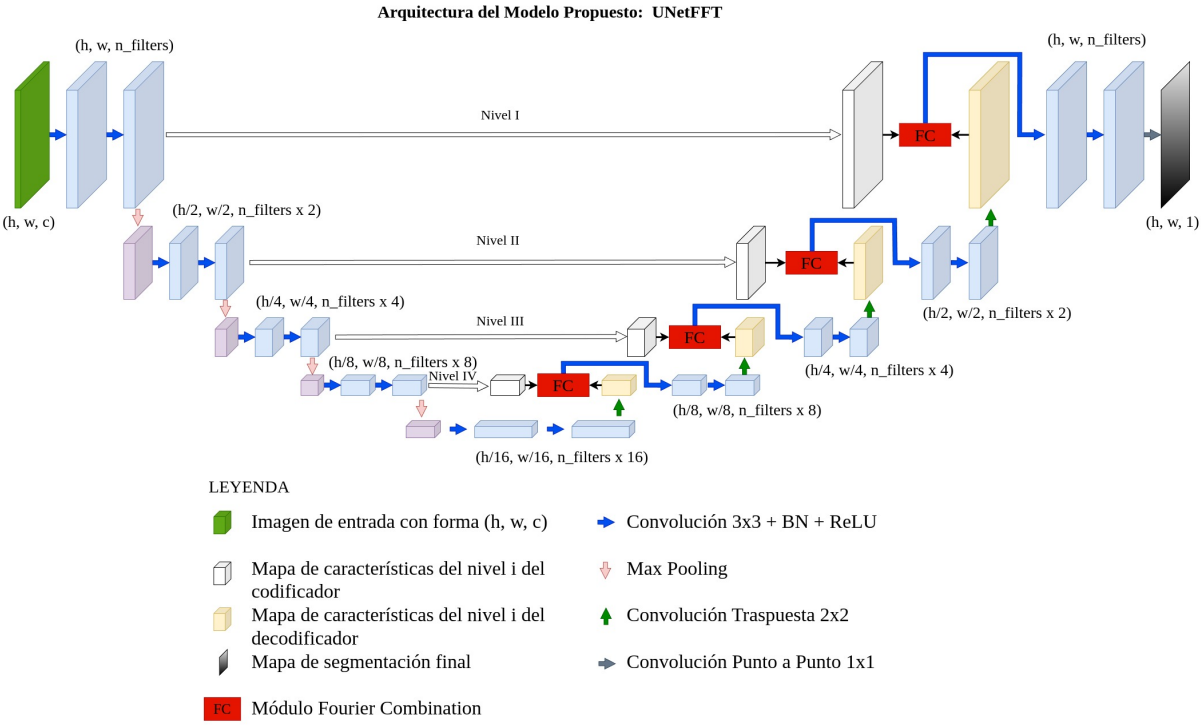
Diagrama de bloques del Módulo FFT diseñado, Fourier Combination.



Nota: Módulo Fourier Combination para la combinación en la frecuencia de los datos del codificador y decodificador en el nivel *i* del modelo.

Figura 3.27

Arquitectura de modelo propuesto UNetFFT.



Nota: Aplica el módulo diseñado, Fourier Combination, para combinar en el dominio de la frecuencia las características del codificador y decodificador.

que la adición de estos cuatro módulos aumenta el número de parámetros de la red y es relativamente sencillo calcularlos: dado que de acuerdo a la Ecuación 3.11, la DFT de una imagen de $M \times N$ crea una representación en la frecuencia con las mismas dimensiones; y que se establecen $n\_filters$ por cada nivel para ajustarse al número de canales del mapa de características, tanto del codificador como decodificador; y sean h y w el alto y ancho (dimensiones espaciales) del mapa de características de entrada y c su respectivo número de canales en el nivel i, el cálculo de la cantidad parámetros aprendibles por nivel del módulo añadido se consigue siguiendo la Ecuación 3.29.

$$\text{\#Parámetros por módulo} = 2 \times h \times w \times c \tag{3.29}$$

Donde el factor 2 se debe a que existen filtros para el mapa de características del codificador y decodificador. Por lo tanto, para el modelo UNetFFT configurado con $n\_filters$ en su primer nivel y que sigue la distribución de tamaños y canales para cada nivel mostrados en la Tabla 3.12, la cantidad de parámetros añadidos al modelo base debido a la adición de los módulos *Fourier Combination*, se puede calcular con las Ecuaciones 3.30, 3.31,3.32,3.33 y 3.34.

$$\text{Nivel I} = 2 \times h \times w \times n\_filters \tag{3.30}$$

$$\text{Nivel II} = 2 \times \frac{h}{2} \times \frac{w}{2} \times (n\_filters \times 2) = h \times w \times n\_filters \tag{3.31}$$

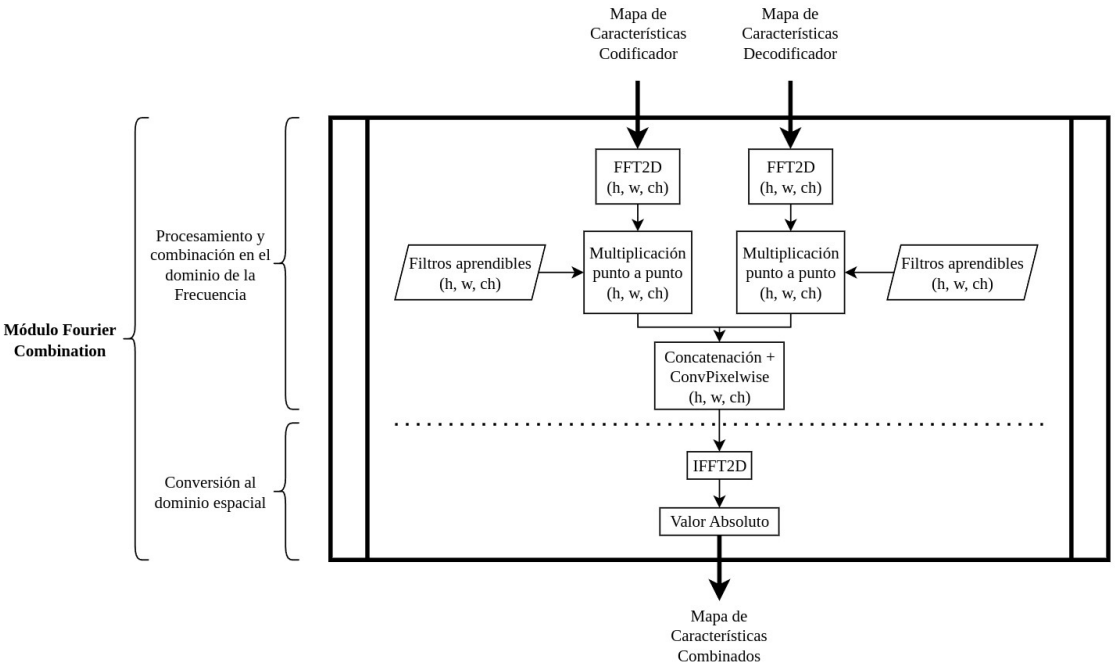
$$\text{Nivel III} = 2 \times \frac{h}{4} \times \frac{w}{4} \times (n\_filters \times 4) = \frac{h \times w \times n\_filters}{2} \tag{3.32}$$

$$\text{Nivel IV} = 2 \times \frac{h}{8} \times \frac{w}{8} \times (n\_filters \times 8) = \frac{h \times w \times n\_filters}{4} \tag{3.33}$$

$$\text{\#Parámetros totales añadidos} = \text{Nivel I} + \text{Nivel II} + \text{Nivel III} + \text{Nivel IV} \tag{3.34}$$

Donde $n\_filters$ es un parámetro configurable de UNetFFT que permite crear instancias del modelo con diferentes números de filtros, permitiendo aumentar o reducir la complejidad y profundidad del modelo. De las Ecuaciones 3.30 - 3.33, se aprecia que el mayor número de parámetros añadidos se encuentra en el nivel I, por lo que para una implementación asequible a nivel de hardware debe considerarse la reducción del tamaño de la imagen de entrada ($h \times w$) o la reducción del número de filtros. Dado que las imágenes de entrada al modelo ya es fijo (determinado por el tamaño de las imágenes del conjunto de datos, esto es, 256×256 píxeles), la consideración debe tomarse en el número de filtros. En la Figura 3.28, se muestra el diagrama de flujo del procesamiento en el módulo Fourier Combination y en la Figura 3.29, se muestra el flujo detallado del modelo propuesto, UNetFFT. La implementación en código del modelo UNetFFT se muestra en el Apéndice C.

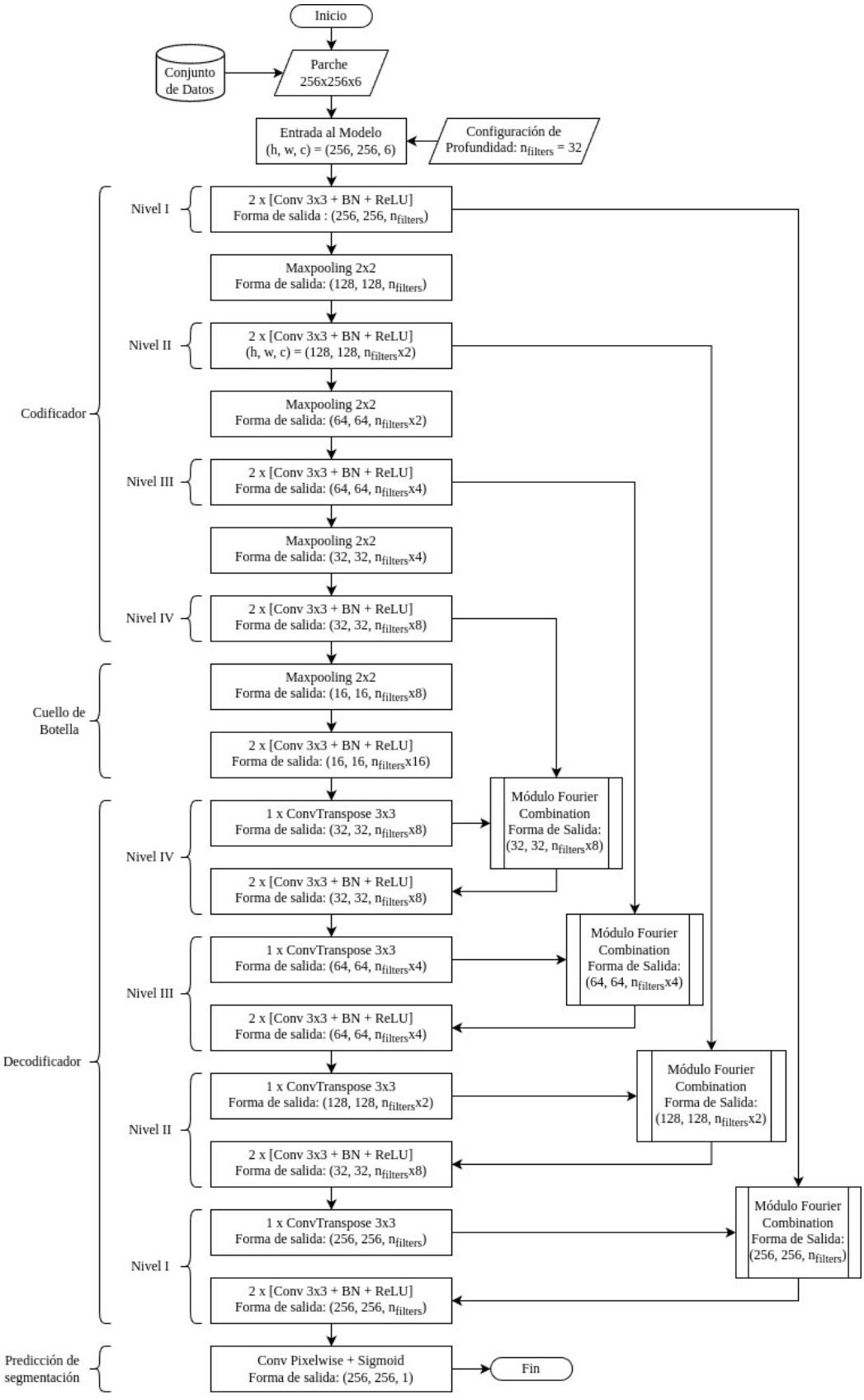
Figura 3.28
Diagrama de flujo del procesamiento del módulo Fourier Combination.



3.3. Métodos para Comparación y Métricas de Evaluación

El modelo UNet original explicado en la sección 3.2.1 es el principal punto de referencia para comparar al modelo propuesto en esta investigación (UnetFFT). Aún así, se utilizan otros

Figura 3.29
Diagrama de flujo de modelo propuesto, UNetFFT.



modelos basados en CNN que son, en general, parte del estado del arte en la segmentación de imágenes (también en teledetección). Estos otros modelos son:

- Fully Convolutional Network (FCN), propuesta por (Long et al., 2015), esta red fue una de las primeras en utilizar solamente capas convolucionales, evitando el uso de arreglos de neuronas o capas densas.
- Linknet, propuesta por (Chaurasia y Culurciello, 2017), busca una implementación eficiente en número de parámetros de una red codificador-decodificador aprovechando sus conexiones directas.
- PSPNet, propuesta por (Zhao et al., 2017), que proponen el módulo *pyramid pooling* para capturar información en un contexto global y a diferentes escalas.

Es una práctica común evaluar y comparar los nuevos modelos con estos que han demostrado importantes resultados en la tarea de segmentación en general.

En el contexto específico del análisis de imágenes satelitales resalta la aplicación del índice NDWI para la detección de cuerpos de agua. Como se mencionó, este método puede llegar a ser altamente preciso aunque con mucho trabajo manual, esto principalmente debido a que este índice confunde con suma frecuencia sombras de nubes y montañas, vegetación y nieve como cuerpos de agua, por lo que una depuración manual de los resultados es necesario, lo que impide una automatización y manejo de grandes cantidades de información, como a menudo es requerido. Aún así, sigue siendo una herramienta utilizada ampliamente junto con otras técnicas como Otsu para la detección automática de umbrales. A continuación se desarrollan los 3 modelos adicionales de comparación así como la metodología del índice NDWI con umbralización por Otsu.

Finalmente, también se desarrolla en esta sección las métricas mediante las cuáles se comparará el rendimiento de las diferentes metodologías.

3.3.1. Fully Convolutional Network

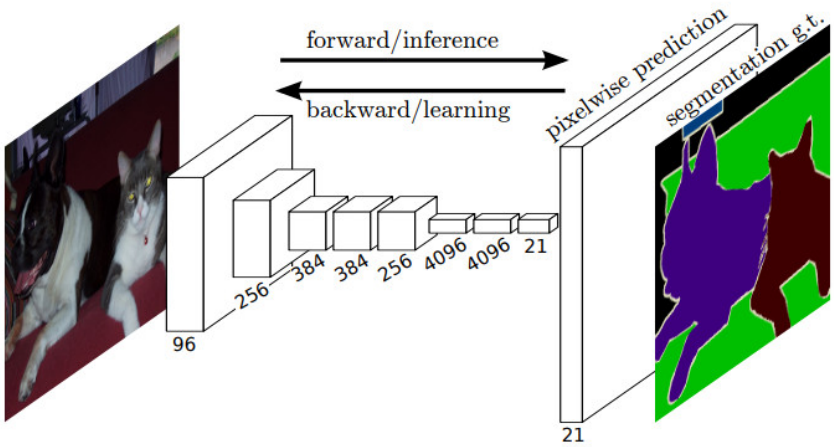
Esta red es unas de las más importantes en el campo del aprendizaje profundo, ya que fue la cambió el paradigma de su momento implementando una red completamente convolucional

(*Fully Convolutional Network*, FCN), esto es, sus predecesoras combinaban redes CNN que se encargaban de abstraer información a diferentes niveles para, finalmente, utilizar capas densas (arreglo de neuronas) en su predicción final. El aporte de esta nueva red fue modificar este último paso, cambiando las capas densas por otras operaciones convolucionales, permitiendo obtener como salida del modelo ya no solo neuronas que clasificaban imágenes, sino obtener una imagen que podría representar la segmentación de algunos elementos dentro ella.

Más allá de su novedad, también obtuvo excelentes resultados, alcanzando el estado del arte para segmentación de imágenes en su momento. Basados en esta nueva perspectiva, nuevos modelos y arquitecturas fueron diseñados.

La implementación original de este modelo consiste en utilizar como columna vertebral la arquitectura VGG16 que era utilizada como extractor de características en modelos de clasificación. Como se mencionó, eliminaron las capas densas de este modelo y la reemplazaron por capas convolucionales traspuestas o deconvoluciones, cuya función es la de aumentar la dimensión espacial de sus datos de entrada, y convoluciones *pixelwise* para la predicción de la máscara final. Una vista general del modelo se puede apreciar en la Figura 3.30.

Figura 3.30
Arquitectura del modelo FCN8: basado en la red VGG16 y deconvoluciones.



Nota: Extraído de (Long et al., 2015).

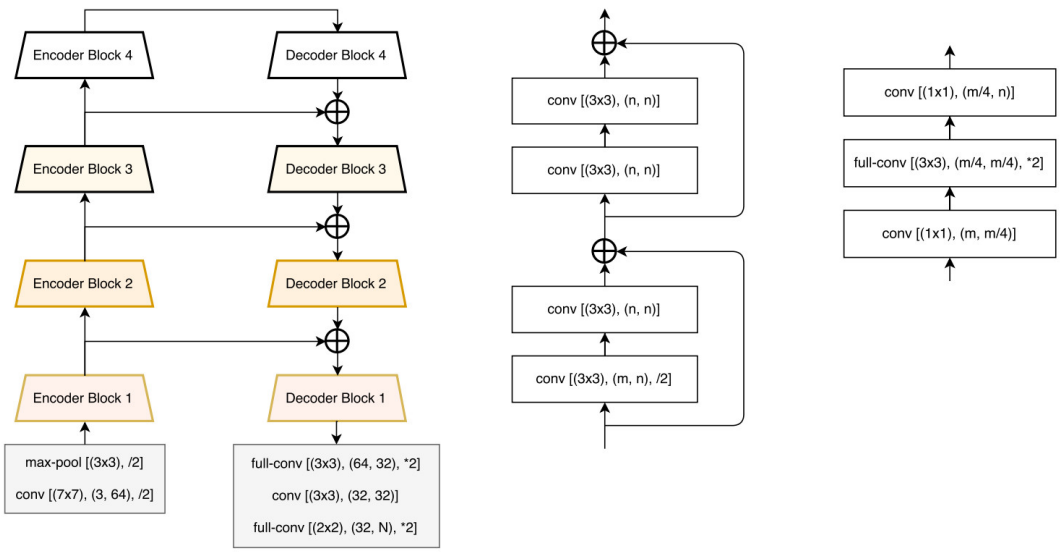
Donde, de la Figura 3.30 se observa que el codificador está basado en VGG16, que reduce las dimensiones espaciales de la imagen de entrada, mientras que aumenta el número de filtros o canales a medida que se profundiza en el codificador. En el decodificador se utilizan capas de deconvolución para aumentar las dimensiones espaciales y se aplican una convolución *pixelwise*

para obtener el mapa de segmentación final. La implementación de este modelo de comparación se realiza siguiendo las consideraciones expuestas en su artículo original (Long et al., 2015) y se muestran en el Apéndice D.

3.3.2. Linknet

La tarea de segmentación se dio en un inicio con el objetivo de tratar con imágenes en tiempo real, sin embargo, la mayoría de modelos de segmentación, aunque con buenos resultados, les tomaba mucho tiempo realizar predicciones por su gran cantidad de parámetros, especialmente en sistemas embebidos, los que tienen limitados recursos. En este contexto, (Chaurasia y Culurciello, 2017) propusieron Linknet, una red de segmentación diseñada para ser liviana y obtener resultados competitivos. Es una red codificador-decodificador, que implementa conexiones residuales en el codificador mientras que aplica convoluciones convencionales y traspuestas en el decodificador. En la Figura 3.31, se observa su arquitectura y la composición de sus bloques.

Figura 3.31
Arquitectura del modelo Linknet.



a) b) c)

Nota: a) Arquitectura del modelo Linknet, b) Bloque codificador, c) Bloque decodificador. Extraído de (Chaurasia y Culurciello, 2017)

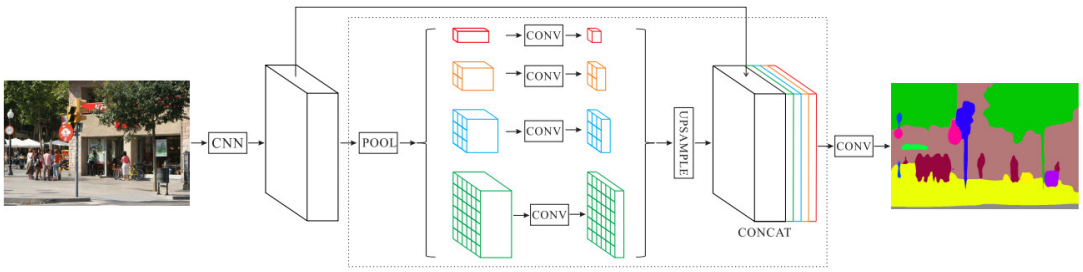
Se observa en la Figura 3.31a) la arquitectura general del modelo, que consiste en el codificador (izquierda) que reduce las dimensiones espaciales y aumenta el número de canales

abstrayendo las características de los datos de entrada, mientras que el decodificador (derecha) combina la información del codificador y aumenta sus dimensiones espaciales; b)representa el bloque codificador que es la aplicación seguida de capas convolucionales 3×3 y que van reduciendo las dimensiones a la mitad por cada bloque, un punto importante es la aplicación de conexiones residuales que evitan el problema de desvanecimiento del gradiente; c) el bloque del decodificador aplica convoluciones para combinar linealmente los datos y reducir y aumentar el número de canales, así como una convolución traspuesta para aumentar el tamaño de la imagen. La implementación del modelo se realiza siguiendo las consideraciones en su artículo original (Chaurasia y Culurciello, 2017) y se muestra el código en el Apéndice E.

3.3.3. PSPNet

Uno de los problemas de las redes CNN clásicas es que a medida que procesan las imágenes, van reduciendo sus dimensiones espaciales. Esto es interesante desde el punto de vista de la abstracción, ya que la información relevante se condensa en imágenes cada vez más pequeñas, sin embargo, cuando se trata del problema de segmentación, la recuperación de estas imágenes pequeñas a las dimensiones originales produce problemas para obtener una segmentación de calidad. Una de las formas de prevenirlo es mediante la aplicación de conexiones de salto, pero no es suficiente, por ello, (Zhao et al., 2017) desarrollaron una nueva arquitectura llamada *pyramid scene parsing network* (PSPNet) basada la aplicación del módulo *pyramid pooling* que utiliza la información a diferentes escalas espaciales y las combina para generar una segmentación con mayor contexto y también mayor abstracción. La arquitectura se muestra en la Figura 3.32.

Figura 3.32
Arquitectura del modelo PSPNet.



Nota: Arquitectura basada en la combinación a diferentes escalas espaciales.

Se observa de la Figura 3.32 que antes de la aplicación del módulo (el bloque en recuadro punteado), se aplica una red CNN. De forma general, esta red podría ser cualquiera que abstraiga la información con CNNs, sin embargo, en la implementación original se utilizó a ResNet50, ampliamente utilizada en tareas de clasificación de imágenes. La salida de ResNet50 es un mapa de características que es la entrada al módulo *pyramid pooling*, su operación básica es reducir en diferentes factores las dimensiones espaciales del mapa de características. La implementación original reducía el mapa de entrada a dimensiones espaciales agrupándolo en grupos de 4, 8, 16 y una agrupación global y aumentando los canales. Dado que cada uno de estos resultados tienen diferentes dimensiones espaciales, se aplica un redimensionamiento mediante interpolación bilineal. Cuando todos tienen la misma dimensión espacial, se concatenan y mediante una convolución *pixelwise*, se obtiene el mapa de segmentación final. La implementación para esta investigación se realiza siguiendo las consideraciones expuestas en su artículo (Zhao et al., 2017) y su implementación en código se muestra en el Apéndice F.

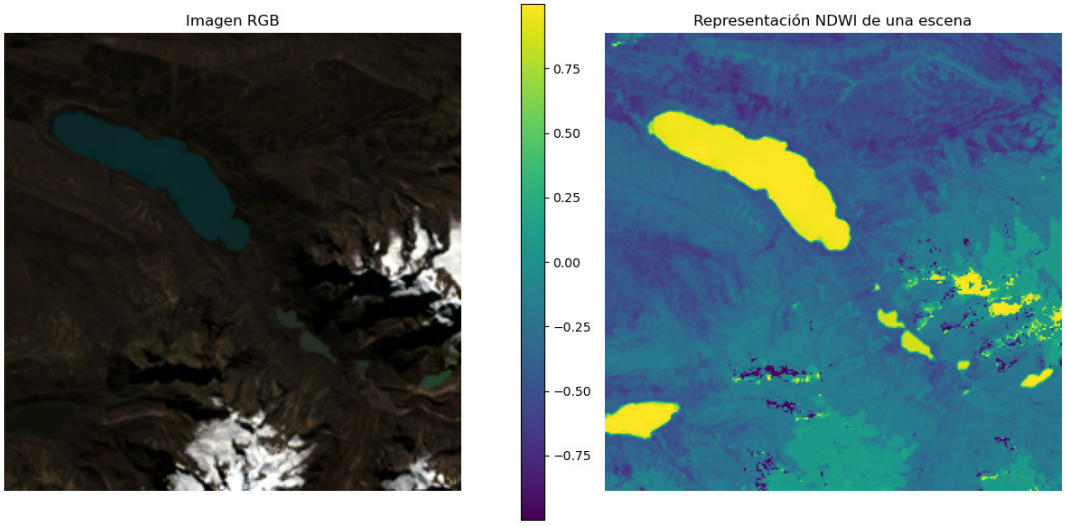
3.3.4. NDWI con Umbralización por Otsu

Desde que (McFeeters, 1996) presentó el índice de agua de diferencia normalizada (NDWI), este método se ha utilizado ampliamente para el delineamiento y segmentación de cuerpos de agua como ríos, lagos y mares. Aunque este método es realmente efectivo detectando cuerpos de agua, también es muy sensible detectando otras entidades como sombras, vegetación y nieve. Esto es especialmente cierto en contextos complejos como en las cordilleras, donde existen múltiples elementos naturales que el índice NDWI identifica como agua, cuando no lo son. La aplicación del índice NDWI produce una imagen de un solo canal cuyos valores oscilan entre -1 y 1, considerándose como laguna cuando el valor de NDWI >0.2 (Ma et al., 2024). Sin embargo, este valor es casi siempre referencial, puesto que, aunque en general puede ser efectivo, en muchas condiciones este umbral tiende a fallar. Por ello, nuevas investigaciones, como la de (Y. Wang et al., 2023), combinan la utilización de NDWI junto con otras técnicas de umbralización dinámica, como el uso del algoritmo de Otsu, que permite la selección automática de un umbral basado en los valores de la escena analizada (umbral adaptativo), obteniendo mejores resultados y de forma automática. Siguiendo la metodología de (Y. Wang et al., 2023), en esta investigación

se utiliza el índice NDWI con umbralización por Otsu como método de comparación.

El índice NDWI utiliza las bandas verde e infrarrojo cercano para su cálculo siguiendo la Ecuación 2.1. El resultado es una imagen de un solo canal, se muestra un ejemplo en la Figura 3.33. Para el ejemplo, el NDWI varía entre -0.8 y 0.8, se observa en color amarillo que se detectan claramente diferentes cuerpos de agua, sin embargo, también se detectan partes del nevado y sombras con el mismo valor de NDWI.

Figura 3.33
Ejemplo de cálculo de índice NDWI.



Nota: A la derecha la representación RGB de una imagen y a la izquierda su cálculo NDWI.

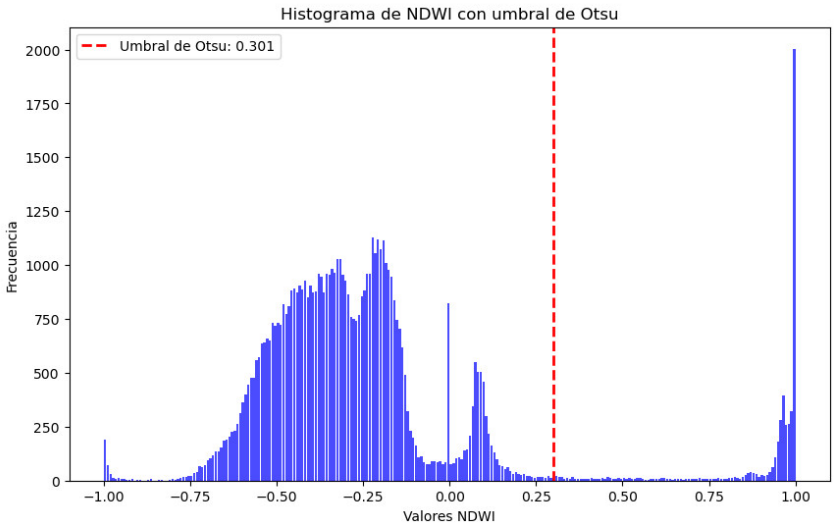
Se aplica el algoritmo Otsu sobre la representación NDWI y se calcula el valor óptimo de NDWI que umbraliza la imagen en dos clases. En la Figura 3.34, se observa cómo se distribuyen los valores de los píxeles de la representación NDWI.

La aplicación del algoritmo Otsu sobre la imagen de ejemplo, devuelve como umbral el valor de 0.301, que separa los dos principales grupos de píxeles. Con este valor se puede binarizar la imagen y obtener la máscara correspondiente a la detección de lagunas. Aunque, como se aprecia en la Figura 3.33, existen entidades que no son agua detectados como si lo fueran.

3.3.5. Métricas de Evaluación

Basados en las investigaciones de (Chen et al., 2023) y (Luo et al., 2021), que implementaron sus propias propuestas de modelos para segmentación de lagunas, se utilizan en esta investigación las métricas *Mean Intersection Over Union*, *F1-Score* y *Pixel Accuracy*.

Figura 3.34
Histograma de NDWI y umbral Otsu.



Mean Intersection over Union (MIoU)

Esta métrica mide el nivel de superposición entre la máscara de predicción y la referencia o valor de verdad. Se calcula mediante la Ecuación 3.35.

$$\text{MIoU} = \frac{1}{k+1} \sum_{i=0}^{k} \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \times 100\,\% \tag{3.35}$$

Donde k representa el número de clases que la predicción realiza, en el caso de esta implementación son dos: laguna y no laguna. Y tal como indica (Chen et al., 2023), esta es la métrica de evaluación más directa para evaluar a los algoritmos en la tarea de segmentación.

F1-Score

Esta métrica es útil para evaluar tareas de clasificación binaria como es la segmentación de dos clases en una imagen. Evalúa la calidad de las máscaras de segmentación producidas por las diferentes metodologías. Se calcula utilizando las métricas de *precision* (3.36) y *recall* (3.37), como se muestra en la Ecuación 3.38.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.36}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.37}$$

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \times 100\,\% \tag{3.38}$$

Pixel Accuracy (PA)

Esta métrica mide el número de píxeles que son clasificados correctamente en la imagen predicha dividido entre el número total de píxeles. Este valor se calcula siguiendo la Ecuación 3.39

$$\text{PA} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \times 100\,\% \tag{3.39}$$

Respecto a las métricas de evaluación definidas, las clasificaciones como verdaderos positivos son representados mediante TP, los verdaderos negativos con TN, los falsos positivos con FP y los falsos negativos con FN.

Capítulo 4

Experimentos y Resultados

En esta sección se describe el entorno en el cual se desarrollan los experimentos, se justifican los parámetros para el entrenamiento de los modelos, se evalúa el comportamiento del modelo propuesto UNetFFT para la tarea de segmentación de lagunas, se compara el resultado cuantitativamente mediante las métricas definidas con otras metodologías. Respecto al modelo propuesto, se comprueba que la inclusión del módulo *Fourier Combination* en el modelo línea de base UNet mejora su rendimiento, siendo esta diferencia significativamente estadística. Finalmente, se observan las características y mejoras del modelo propuesto de forma cualitativa.

4.1. Configuraciones Experimentales

4.1.1. Entorno de Trabajo

Todos los experimentos descritos en esta sección se realizaron, a nivel de hardware, en una estación de trabajo con un procesador Intel Xeon W-2123, 32 GB de memoria RAM y una tarjeta gráfica NVIDIA Quadro P2000 con 5GB de memoria. A nivel de software, todos los códigos son escritos en Python y usando Tensorflow 2.15.0 como *framework* de aprendizaje profundo, ambos sobre un sistema operativo Debian GNU/Linux 12.

4.1.2. Parámetros de Entrenamiento de los Modelos

Con la finalidad de realizar un comparación objetiva y justa entre los diferentes modelos, todos se entrenan bajo las mismas condiciones y parámetros. Siguiendo la implementación de (Luo et al., 2021), los modelos se compilan utilizando el algoritmo Adam (Kingma, 2014) como optimizador, configurado por defecto con una tasa de aprendizaje de 0.001. Dado que el objetivo en la investigación es la segmentación binaria, la función de pérdida utilizada es la entropía cruzada binaria (*binary cross-entropy*), definida mediante la Ecuación 4.1.

$$L(y_i, \hat{y_i}) = -\frac{1}{N}\sum_{i=0}^{N}(y_i \log \hat{y_i}) + (1 - y_i)\log(1 - \hat{y_i}) \tag{4.1}$$

Esta función de pérdida recorre los N píxeles de la imagen y calcula la entropía cruzada binaria de cada píxel i y los promedia.

El modelo se entrena durante 200 épocas y se realiza el seguimiento basado en la métrica MIoU. Se implementa un control de sobreajuste mediante la configuración de una parada anticipada (*early stopping*) configurada en 15, que indica que si en 15 épocas la métrica de evaluación no mejora, el entrenamiento se detiene, evitando que el modelo memorice la información de entrada. El tamaño de lote que los modelos procesan de forma paralela depende de la capacidad de memoria del equipo donde se realizan los experimentos, dadas las condiciones de hardware, experimentalmente se comprueba que el tamaño de lote adecuado es de 4 muestras por paso. La forma de entrada a todos los modelos es $(256, 256, 6)$ que representa una imagen de 256×256 píxeles y 6 canales. Todos estos parámetros son resumidos en la Tabla 4.1.

Tabla 4.1
Parámetros de entrenamiento para los modelos.

Ítem	**Valor del parámetro**
Forma de entrada	$256 \times 256 \times 6$
Número de clases predichas	2
Épocas	200
Parada anticipada	Después de 15 épocas.
Optimizador	Adam
Tasa de aprendizaje	0.001
Función de pérdida	Entropía cruzada binaria
Tamaño de lote	4

El código para entrenar los modelos con los parámetros indicados se muestra en el Apéndice G.

4.1.3. Aumento de Datos para el Entrenamiento

Todos los modelos son entrenados con el conjunto de datos elaborado y descrito en la Sección 3.1. Se utiliza específicamente el subconjunto de entrenamiento y validación junto con procedimientos de aumento de datos, cuyo objetivo es mejorar la generalización y prevenir el sobreajuste del modelo, los cuales son especialmente importantes cuando el conjunto de datos es limitado. Siguiendo la implementación de (Luo et al., 2021), el aumento de datos consiste en la aplicación de rotaciones, reflexiones horizontales y verticales, así como la adición de ruido gaussiano, cada proceso con una probabilidad del 50 % de aplicarse cada vez que se introduce una imagen a los modelos. De acuerdo a la Tabla 3.10, el entrenamiento se realiza con 560 imágenes que pasan por los procesos de aumento de datos mencionados y son entregadas al modelo. En cada época se evalúa el rendimiento del modelo con las 80 imágenes de validación que los modelos no vieron durante su entrenamiento sobre la métrica de MIoU. El seguimiento entre de la métrica de evaluación en el entrenamiento y la validación permite saber cuándo el modelo se está sobreajustando.

4.1.4. Especificaciones de los Modelos Implementados

En la Sección 3.2 se presentó de manera general la arquitectura del modelo propuesto, UNetFFT, el cual permite configurar diferentes profundidades ajustando el parámetro *n_filters*. Para esta investigación, se han instanciado dos variantes del modelo: a) UNetFFT16, con $n\_filters = 16$ y b) UNetFFT32, con $n\_filters = 32$.

Similarmente, para el modelo línea de base UNet, sigue esta misma configuración de profundidad basada en $n\_filters$, se instancian 3 modelos de UNet: a) UNet16, b) UNet32 y c) UNet64, siendo este último la implementación clásica de este modelo.

Se sigue un proceso similar con el modelo FCN, ajustando la profundidad de su red codificadora basada en VGG16 mediante la configuración de 32 y 64 filtros en la capa de entrada, lo que da lugar a dos variantes: a) FCN32 y b) FCN64. De igual manera, se procede con los

modelos Linknet y PSPNet, generando las variantes c) Linknet32, d) Linknet64, e) PSPNet32 y f) PSPNet64.

Los modelos implementados se resumen en la Tabla 4.2.

Tabla 4.2
Modelos implementados para experimentación.

Nombre del modelo	Filtros de entrada
UNetFFT16	16
UNetFFT32	32
UNet16	16
UNet32	32
UNet64	64
FCN32	32
FCN64	64
Linknet32	32
Linknet64	64
PSPNet32	32
PSPNet64	64

4.2. Entrenamiento de los Modelos

Siguiendo las configuraciones explicadas en la Sección 4.1, se entrenan todos los modelos de la Tabla 4.2. De acuerdo a (Florez et al., 2023), cada modelo se entrena 10 veces, se evalúa el modelo sobre el subconjunto de datos de evaluación en las métricas MIoU, F1-Score y PA para cada entrenamiento y se calcula el valor promedio y la desviación estándar. Adicionalmente, se calculan las métricas para la metodología NDWI con umbralización por Otsu. Estos resultados se muestran en la Tabla 4.3.

De la Tabla 4.3, se observa que el modelo UNetFFT32 obtuvo el mejor desempeño en todas las métricas, seguido de UNetFFT16, ambas siendo instancias de la arquitectura propuesta en esta investigación. Además, se observa que la desviación estándar de las métricas en los 10 entrenamientos es la más baja, lo que indica que tanto UNetFFT16 como UNetFFT32 son consistentes y estables en sus procesos de entrenamiento.

El modelo más liviano es UNet16 con 2.16 millones de parámetros, siguiéndole Linknet32 con 2.84 millones. Ambos modelos tienen métricas menores a los mejores, además, sus valores

Tabla 4.3
Métricas de evaluación de los modelos entrenados.

Modelo	MIoU (%)	F1-Score (%)	PA (%)	Parám. (M)
NDWI + Otsu	46.33	26.45	70.41	-
FCN32	82.70 ± 3.83	76.60 ± 7.11	99.59 ± 0.10	7.57
FCN64	83.37 ± 2.03	78.44 ± 2.39	99.63 ± 0.07	30.26
Linknet32	82.07 ± 3.36	75.51 ± 5.89	99.57 ± 0.10	2.84
Linknet64	82.49 ± 1.31	76.21 ± 2.14	99.58 ± 0.04	11.30
PSPNet32	80.87 ± 4.71	73.39 ± 9.22	99.53 ± 0.14	11.77
PSPNet64	80.96 ± 2.91	73.79 ± 5.02	99.53 ± 0.11	46.94
UNet16	81.76 ± 2.57	74.35 ± 4.56	99.60 ± 0.07	**2.16**
UNet32	82.44 ± 3.36	75.38 ± 5.77	99.62 ± 0.09	8.64
UNet64	79.27 ± 4.98	69.29 ± 9.12	99.51 ± 0.19	34.53
UNetFFT16	85.40 ± 1.12	80.69 ± 1.74	**99.67 ± 0.04**	8.4
UNetFFT32	**85.94 ± 1.38**	**81.60 ± 2.09**	99.67 ± 0.05	33.6

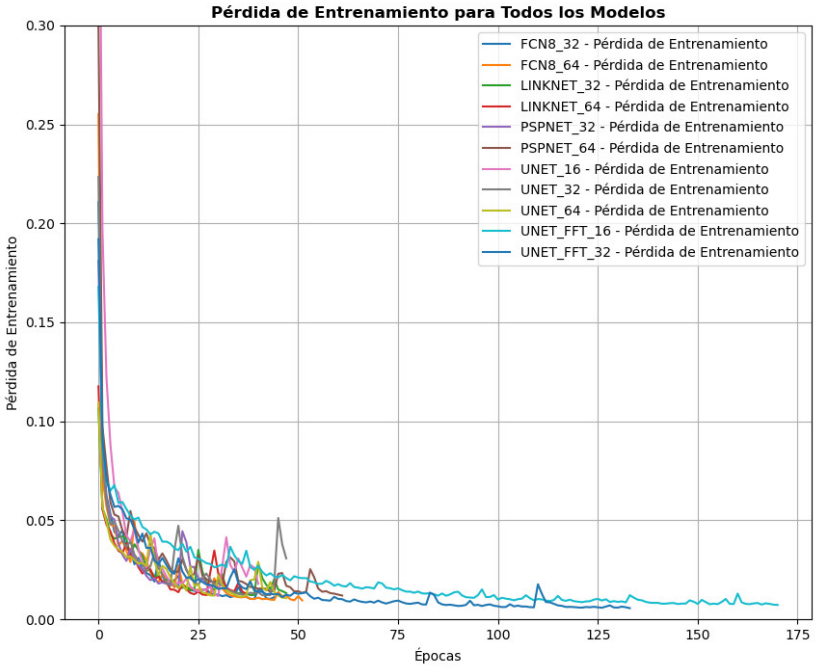
de desviación estándar son mayores, lo que significa que los modelos son menos estables debido a parámetros aleatorios introducidos.

En la Figura 4.1 se observa cómo varía el valor de la pérdida por época para el entrenamiento y validación. De las 10 iteraciones de entrenamiento, se toma aquella en la que cada modelo obtuvo los mejores resultados para generar la gráfica.

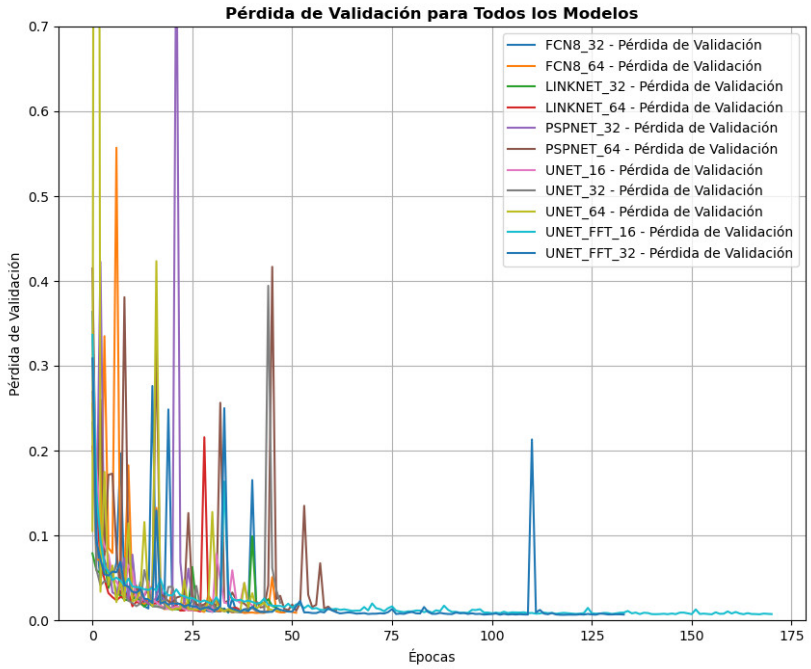
Se observa en la Figura 4.1a) los valores de la pérdida en el proceso de entrenamiento de todos los modelos. Es evidente, cómo a medida que las épocas avanzan los modelos van obteniendo pérdidas menores. Dada la configuración de parada anticipada, se observa que no todos los entrenamientos llegan a las 200 épocas configuradas. Se observa que los modelos de comparación obtienen sus mejores resultados entre las 30 y 50 épocas de entrenamiento, mientras que el modelo propuesto instanciado en UNetFF16 y UNetFFT32, toma muchas más épocas en llegar a su menor valor (entre 130 y 175 épocas). Sin embargo, por simple inspección visual, se observa que las instancias del modelo propuesto tienen muchas menos variaciones abruptas en todo su proceso de entrenamiento, hecho que contrasta con los múltiples picos en la función de pérdida de los otros modelos de comparación.

Respecto a la Figura 4.1b), la cual representa el valor de pérdida evaluado en el conjunto de datos de validación. En este caso, la función de pérdida para los modelos de comparación es bastante inestable, hecho evidente en los múltiples picos y de alto valor que se observan. Por otro lado, tanto los modelos UNetFFT16 y UNetFFT32, son más estables y se corresponden en

Figura 4.1
Valores de pérdida de los modelos entrenados durante el a) entrenamiento y b) validación.



a)

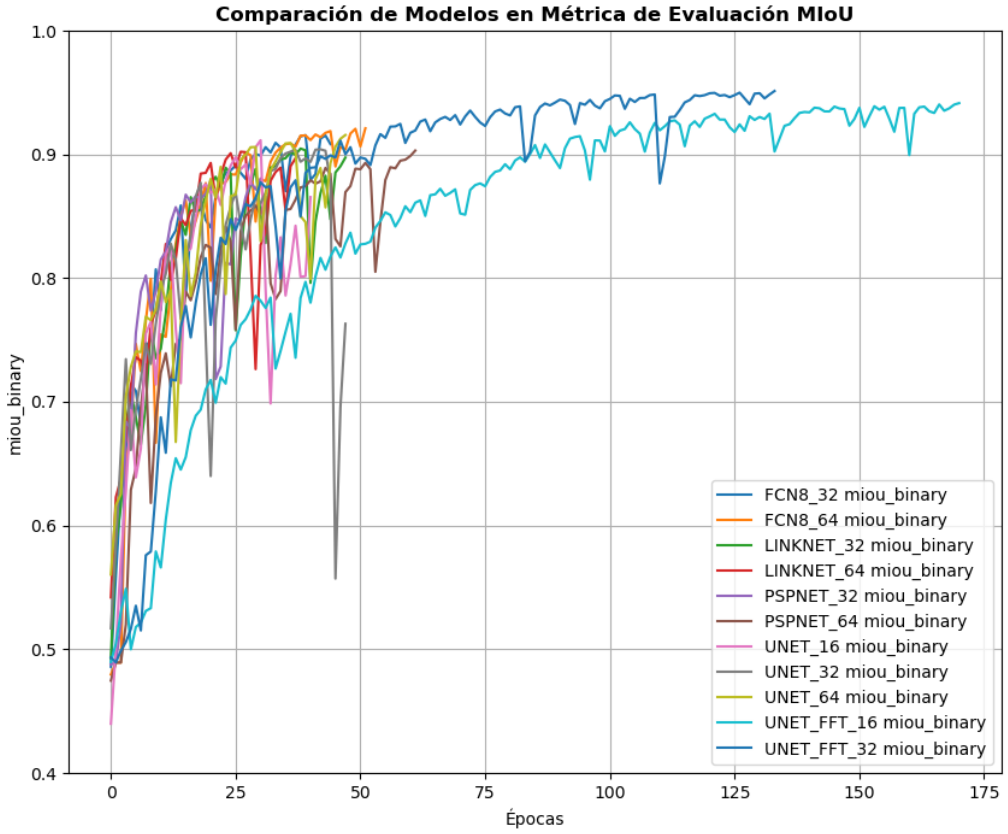


b)

comportamiento con sus respectivos valores de entrenamiento. Comparando estos dos últimos modelos, es de notar que UNetFFT16 es más estable, ya que no muestra picos en los valores de pérdida, mientras que los de UNetFFT32 contiene ciertos picos que reflejan cierta inestabilidad.

En la Figura 4.2 se muestra el valor de la métrica MIoU para el conjunto de datos de validación.

Figura 4.2
Gráfica de MIoU en el conjunto de datos de validación.



De acuerdo a la Figura 4.2, se observa que, en general, los modelos de comparación convergen más rápidamente hacia sus valores máximos de MIoU, sin embargo, ya no mejoran conforme más épocas se entrenan, además, sus resultados son bastante inestables. Similarmente a los valores de pérdida, para los modelos UNetFFT16 y UNetFFT32 la métrica MIoU le toma más épocas llegar a su valor máximo, aunque ambos modelos obtienen los mejores valores de esta métrica sin el problema de sobreajuste. Se observa que el modelo UNetFFT32 obtiene un valor más alto de métrica MIoU y en menos épocas respecto a UNetFFT16, además converge más rápidamente a su valor más alto. Las dos instancias del modelo propuesto tienen más estabilidad en sus predicciones.

En términos generales y de acuerdo a la Tabla 4.3 y a las Figuras 4.1 y 4.2, es evidente que

el modelo propuesto UNetFFT, con sus dos instancias UNetFFT16 y UNetFFT32, es mejor en la tarea de segmentación de lagunas que los modelos de comparación utilizados, obteniendo hasta un 6 % de mejora en la métrica MIoU, aunque requiere más épocas para su entrenamiento y converge más lentamente.

4.3. Análisis Cualitativo de las Predicciones de los Modelos

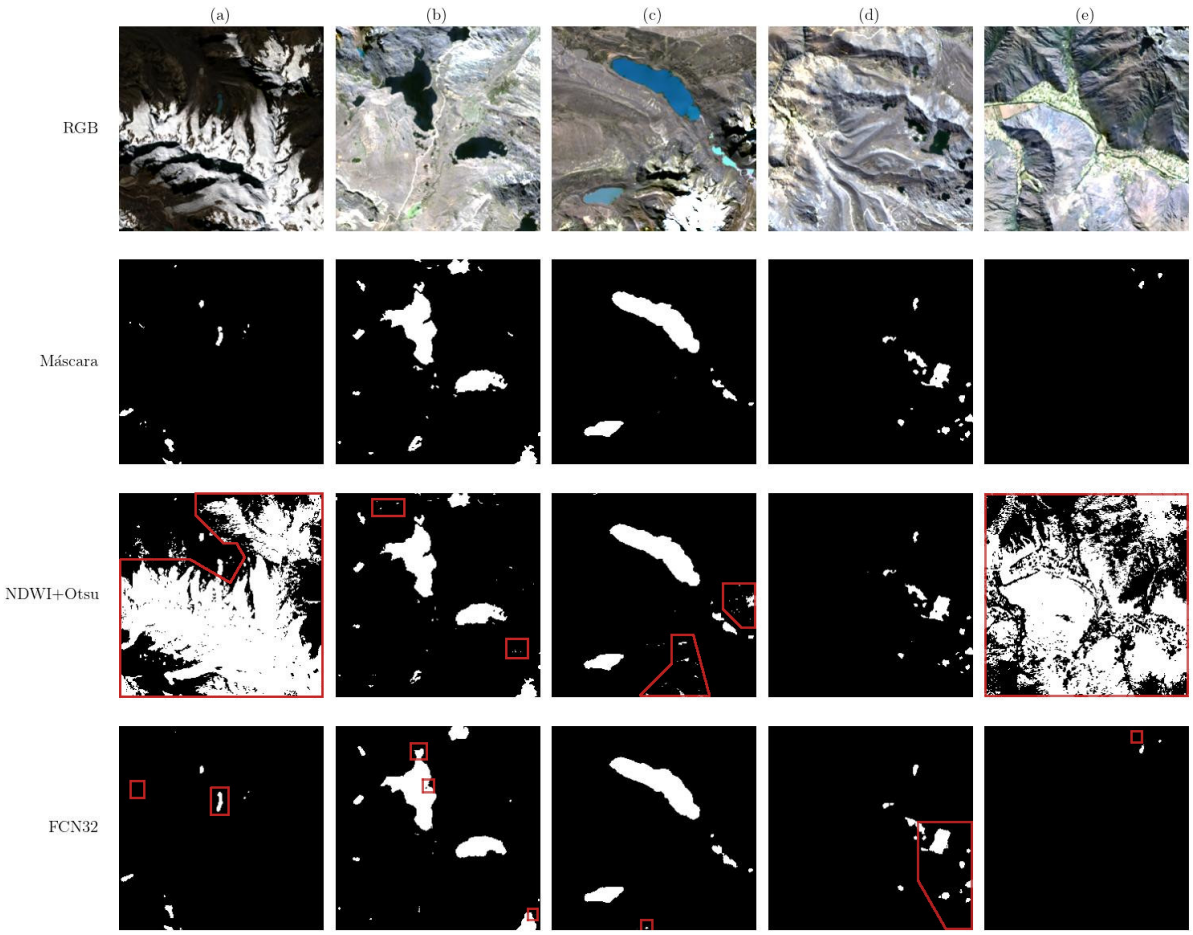
Se evalúa cualitativamente el comportamiento de los modelos entrenados tomando 5 imágenes del conjunto de datos de evaluación que muestran contextos diferentes. En las Figuras 4.3, 4.4 y 4.5 se aprecian la representación RGB de las imágenes, las máscaras y las predicciones realizadas por cada uno de los modelos y metodologías.

De las Figuras 4.3, 4.4 y 4.5, se observa que la columna (a) representa una zona con alta presencia de nevados y muchas sombras debido a las cordilleras, en este contexto, se aprecia que la metodología que utiliza NDWI junto con la umbralización automática por Otsu obtiene resultados pésimos visualmente detectables. La mayor parte de la zona nevada, incluyendo sombras son detectadas erróneamente como "lagunas". Sin embargo, las lagunas verdaderas sí son correctamente clasificadas. Para esta escena, los modelos FCN32, FCN64, Linknet32, Linknet64, PSPNet32 y PSPNet64 tienen problemas para detectar ciertas lagunas de pequeñas dimensiones, además que delinean de forma imprecisa los bordes; a pesar de ello, no confunden nieve ni sombras como cuerpos de agua. Los modelos UNet16, UNet32 y UNet64 tienen los problemas de detección de lagunas pequeñas, aunque delinean de mejor manera los bordes. Los modelos propuestos, UNetFFT16 y UNetFFT32 tienen una gran capacidad de delineamiento de los límites de las lagunas, además que son más precisos incluso identificando lagunas pequeñas.

Respecto a la columna (b), esta representa dos lagunas de gran extensión rodeadas de otras más pequeñas, así como una laguna superficial con presencia de vegetación acuática. En este contexto, la segmentación realizada mediante el índice NDWI con umbralización Otsu es bastante preciso para delinear los cuerpos de agua, sin embargo, clasifica erróneamente algunas sombras como lagunas, así también, tiene dificultades en la segmentación de lagunas superficiales con alta presencia de vegetación. Respecto a los modelos FCN32, FCN64, Linknet32,

Figura 4.3

Análisis cualitativo de los modelos entrenados. Parte 1.



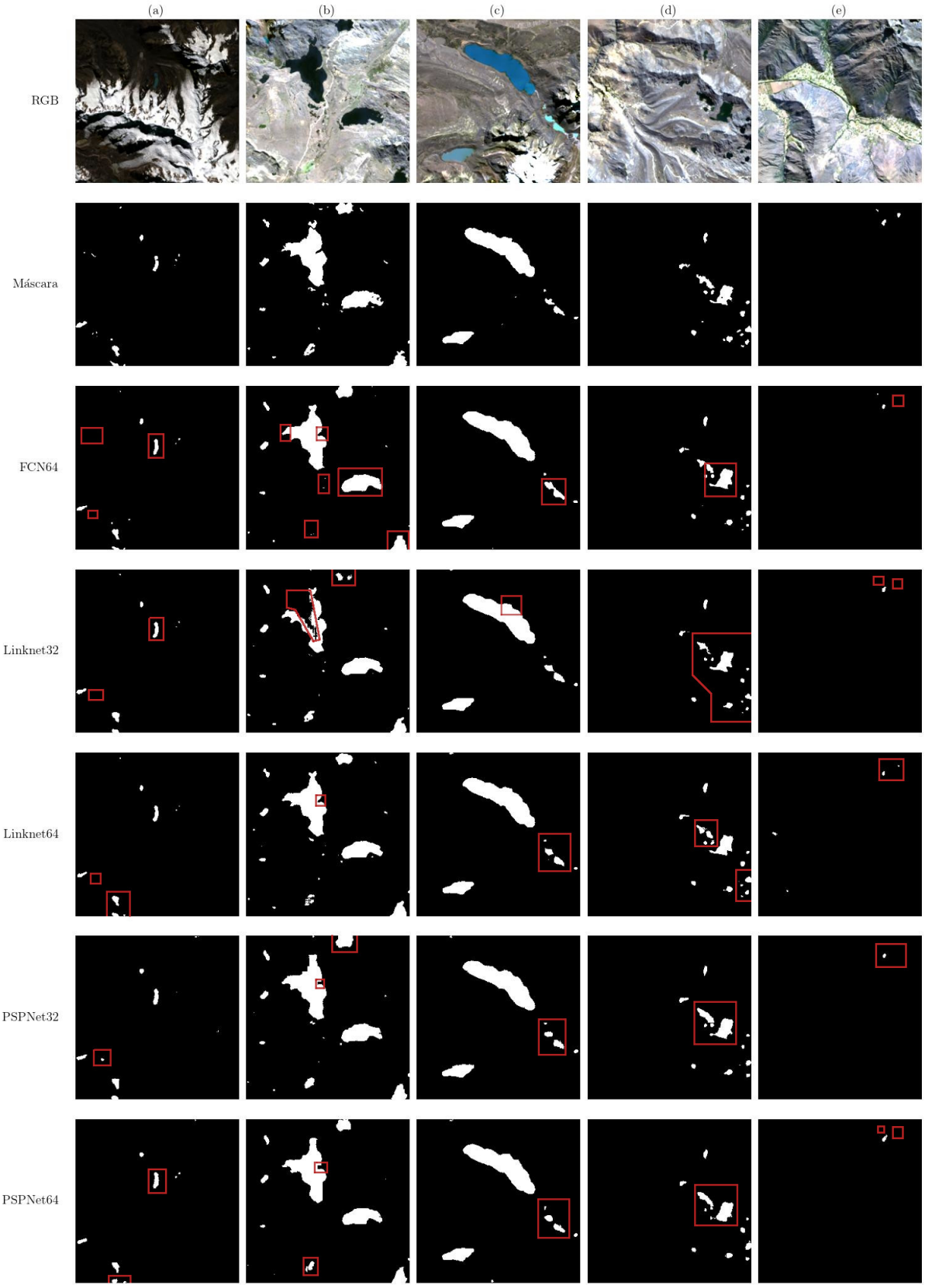
Nota: Representa 5 imágenes en diferentes contextos. Muestra la representación RGB, las máscaras y los resultados de la metodología NDWI con umbralización Otsu y el modelo FCN32.

Linknet64, PSPNet32 y PSPNet64, se aprecia que tienden a detectar de forma imprecisa los bordes de las lagunas y tienen ciertos problemas en la segmentación de lagunas superficiales, siendo Linknet32 quien falló incluso en la detección precisa de la laguna más grande. Finalmente, los modelos UNet16, UNet32 y UNet64, segmentaron mejor los cuerpos de agua, pudiendo incluso detectar pequeñas separaciones entre dos cuerpos de agua contiguos. Los modelos UNetFFT16 y UNetFFT32 obtuvieron segmentaciones bastante más precisas en general, detectando incluso las lagunas pequeñas y dispersas, sin embargo, ambos modelos dificultaron en la segmentación de la laguna superficial con presencia de vegetación acuática.

Sobre la columna (c), esta muestra dos lagunas amplias con forma regular así como otras tres pequeñas bastante juntas. La metodología NDWI con Otsu obtuvo resultados precisos para las

Figura 4.4

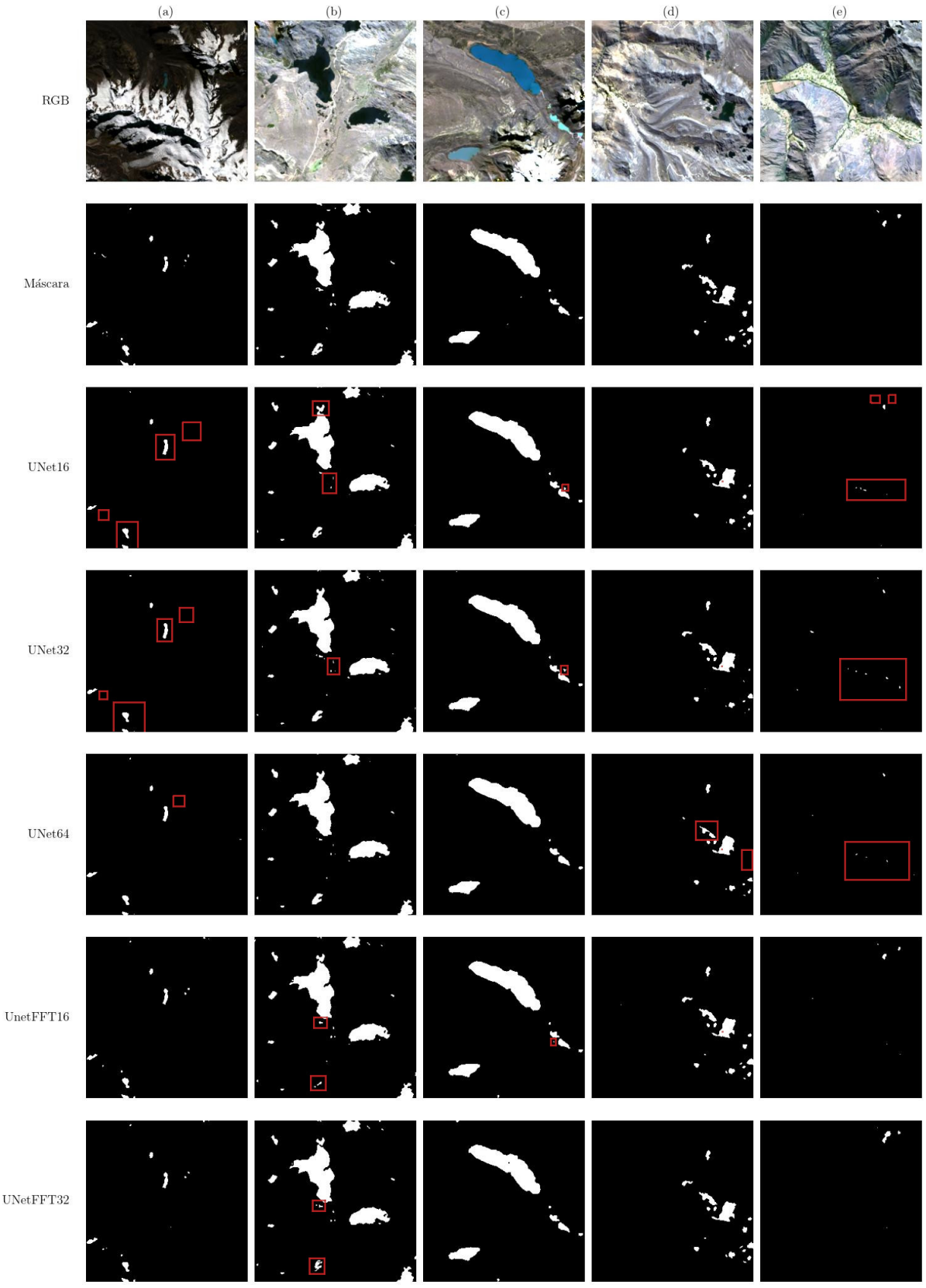
Análisis cualitativo de los modelos entrenados. Parte 2.



Nota: Representa 5 imágenes en diferentes contextos. Muestra la representación RGB, las máscaras y las predicciones de los modelos FCN64, Linknet32, Linknet64, PSPNet32 y PSPNet64.

Figura 4.5

Análisis cualitativo de los modelos entrenados. Parte 3.



Nota: Representa 5 imágenes en diferentes contextos. Muestra la representación RGB, las máscaras y las predicciones de los modelos UNet16, UNet32, UNet64, UNetFFT16 y UNetFFT32.

lagunas, aunque errados para algunas sombras y parte de nevado. Vuelve a ser evidente la falta de delineamiento de bordes de los modelos FCN32, FCN64, Linknet32, Linknet64, PSPNet32 y PSPNet64, especialmente en cuerpos pequeños y bastante juntos. Finalmente, tanto UNet16, UNet32, UNet64, UNetFFT16 y UNetFFT64 muestran resultados de segmentación bastante precisos en este contexto.

En la columna (d) se muestran pequeñas lagunas dispersas en un contexto montañoso y árido sin presencia de nieve. En esta muestra, la segmentación con NDWI produce resultados bastante precisos, sin confundir los elementos del fondo con lagunas. Los modelos de comparación FCN32, FCN64, Linknet32, Linknet64, PSPNet32 y PSPNet64 en general pueden identificar la mayoría de las lagunas aunque tiene dificultades con las pequeñas y se resalta la problemática de los bordes no definidos. Los modelos UNet16, UNet32 y UNet64 se segmentan de forma bastante precisa estas lagunas, aunque alguno de sus bordes no son tan similares a la máscara o etiqueta. Las instancias del modelo propuesto, UNetFFT16 y UNetFFT32, obtienen los mejores resultados de segmentación, tanto en detección de pequeñas lagunas como en clasificación fina de los bordes.

Por último, la columna (e) representa una escena sin casi ninguna laguna, un entorno montañoso y atravesado por un río. En este caso, los modelos de comparación tienen problemas detectando las pequeñas lagunas, mientras que identifican algunos píxeles pertenecientes al río como lagunas, lo cual es errado. En este contexto, las instancias del modelo propuesto tienen una mejor predicción en la segmentación incluso cuando la escena no contiene lagunas.

Del análisis cualitativo, se evidencia que la segmentación mediante el índice NDWI con umbralización por Otsu es el más susceptible a diferentes contextos, siendo especialmente impreciso en escenas donde existe presencia de nieve y sombras (Figura 4.3 (a)), así como en escenas donde las lagunas son una clase muy minoritaria (Figura 4.3 (e)). En general, los modelos de comparación no tienen problemas diferenciando lagunas del fondo (sombras, nieve, vegetación), aunque sus principales limitaciones son la detección de lagunas pequeñas y detalles finos sobre todo en los bordes de las lagunas. Se evidencia también que las instancias del modelo propuesto son bastante precisos en la detección de lagunas y que detectan los bordes de una forma más exacta, identificando incluso lagunas pequeñas y detalles finos, superando a

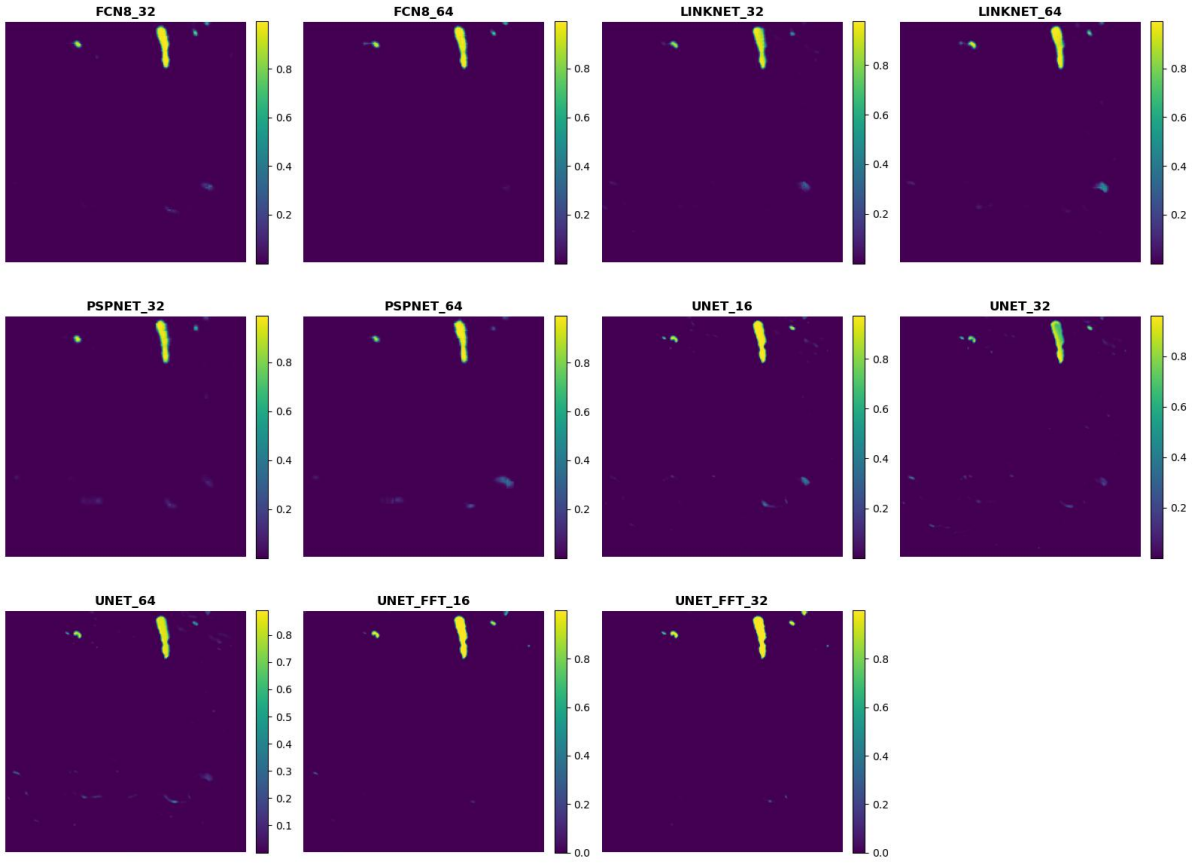
los demás modelos, reflejando en este análisis los resultados de las métricas calculadas en la Sección 4.2.

4.4. Análisis en la Calidad de Segmentación de Bordes

De acuerdo a (X. Zhang et al., 2022), la aplicación directa de los modelos existentes basados en CNN para la segmentación de cuerpos de agua producen detección de bordes borrosos. En este experimento se toman las predicciones sin ningún procesamiento adicional para una muestra del conjunto de datos de evaluación y se muestran sus histogramas correspondientes.

En la Figura 4.6 se muestran las predicciones de cada uno de los modelos entrenados con sus valores sin procesar, esto para visualizar cómo se distribuyen los valores de predicción en los bordes de las lagunas.

Figura 4.6
Predicciones sin procesamiento para cada uno de los modelos.



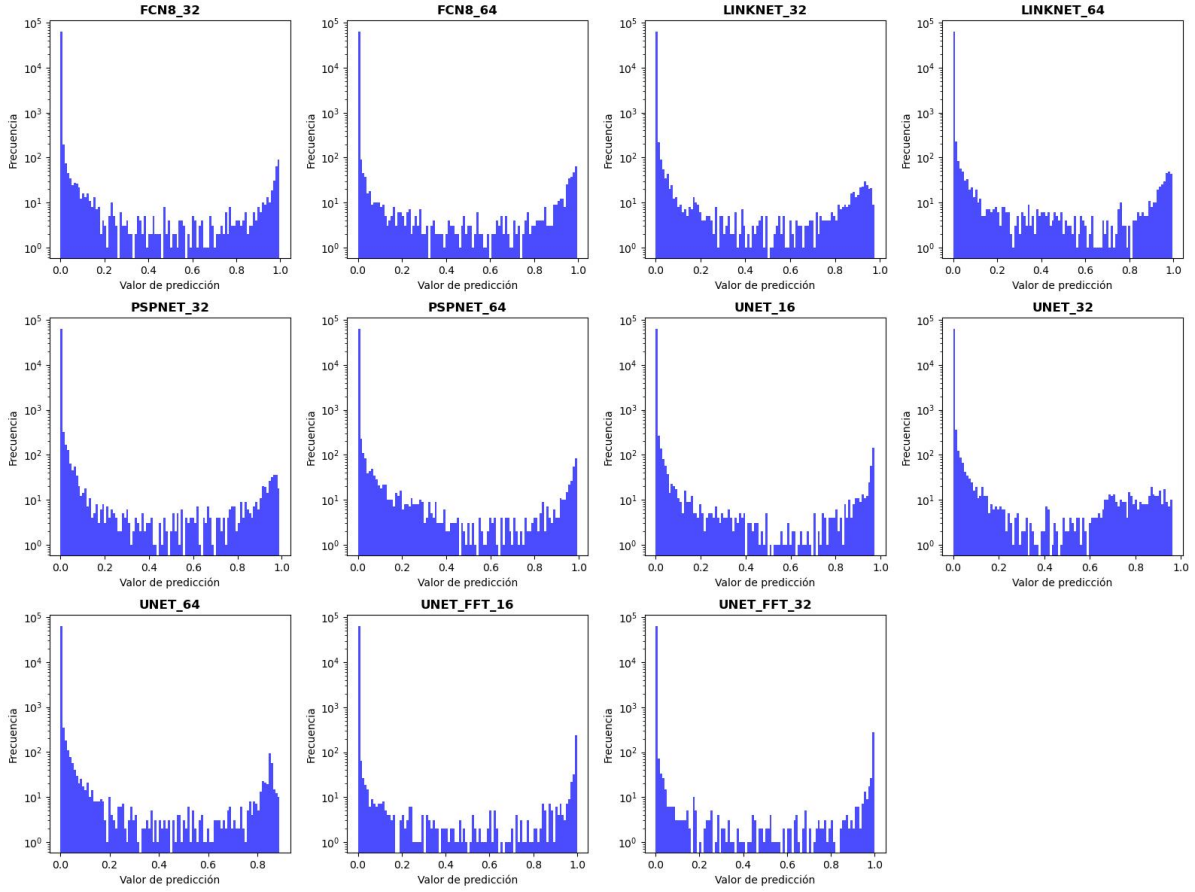
Los modelos de segmentación binaria predicen la probabilidad de que un píxel sea conside-

rado o no como determinado elemento. Esta probabilidad se representa como un valor entre 0 y 1. Se observa de la Figura 4.6 que los modelos de comparación FCN32, FCN64, Linknet32, Linknet64, PSPNet32, PSPNet64, UNet16, UNet32 y UNet64 son relativamente buenos diferenciando el fondo de las lagunas, sin embargo, tienen cierta dificultad definiendo los límites de los cuerpos de agua, dando valores de probabilidad menores a 0.7 para los píxeles próximos a los bordes de las lagunas. Por otro lado, las instancias de los modelos propuestos tienden a predecir con alto nivel de probabilidad cada píxel como perteneciente a lagunas, mostrándose segmentaciones más precisas especialmente en los bordes. En específico, el modelo UNetFFT32 muestra el nivel más alto en la detección de bordes.

En la Figura 4.7, se muestra la representación en histograma de las predicciones anteriores.

Figura 4.7

Histograma de las predicciones sin procesamiento para cada uno de los modelos.



De la Figura 4.7 se observa que los modelos de comparación (FCN32, FCN64, Linknet32, Linknet64, PSPNet32, PSPNet64, UNet16, UNet32, UNet64) dan probabilidades de predicción con valores que oscilan entre 0 y 1, con bastantes datos ubicados en el rango entre 0.2 y 0.8, lo

que indica que estos modelos no son capaces de diferenciar con un alto nivel de probabilidad si un píxel es parte o no de una laguna. Por otro lado, tanto el modelo UNetFFT16 y UNetFFT32 tienen su valores de predicción más separados en dos grupos, uno alrededor del valor de 0 (0 % de probabilidad de que el píxel sea parte de una laguna) y alrededor de 1 (100 % de probabilidad de que el píxel sea parte de una laguna). Existen valores intermedios, pero estos son bastante menos en comparación con los otros modelos. También se evidencia que los modelos propuestos predicen algunos píxeles como 100 % lagunas, mientras que sus contrapartes llegan la mayoría de veces a predecir si un píxel es laguna apenas con un 95 % de probabilidad.

Esto indica que el modelo propuesto es más preciso diferenciando lagunas de su fondo, además que es mucho más preciso en la detección de bordes y formas más finas.

4.5. Estudios de Ablación

Los estudios de ablación en el campo del aprendizaje profundo consisten en evaluar cómo impacta la modificación de un modelo en sus resultados. Para esta investigación, como se indicó en la Sección 3.2.1, el modelo línea de base es el modelo UNet. Su implementación original consideraba $n\_filters = 64$ filtros de entrada. En base a este modelo, se añade el módulo **Fourier Combination** que busca mejorar la precisión de las predicciones, lo que se pudo observar tanto cuantitativa como cualitativamente en las Secciones 4.2, 4.3 y 4.4. Sin embargo, como indican (Rainio et al., 2024), se pueden utilizar pruebas estadísticas para estimar si los diferentes valores en las métricas de dos modelos se deben a diferencias reales entre ellos, recomendando la utilización de la prueba de rangos con signo de Wilcoxon para comparar dos modelos.

Para este estudio, como se detalló en la Tabla 4.2, se implementaron dos instancias del modelo propuesto, a saber: UNetFFT16 y UNetFFT32, cada uno con 16 y 32 filtros en el primer nivel, respectivamente. No se instanció el modelo UNetFFT64 debido a que la cantidad de parámetros de entrenamiento de este modelo excedían el hardware disponible. Por ello, el estudio de ablación se realiza comparando UNet16 con UNetFFT16 y UNet32 con UNetFFT32, evaluando estadísticamente el impacto de la adición del módulo basado en la transformada de

Fourier en el modelo línea de base.

Siguiendo la metodología de (King y Eckersley, 2019), se formula la hipótesis nula y alternativa, indicadas en las Ecuaciones 4.2 y 4.3, respectivamente.

$$H_0 : mediana(D) = 0 \tag{4.2}$$

$$H_a : mediana(D) \neq 0 \tag{4.3}$$

Donde D representa las diferencias entre los valores de MIoU de cada imagen del conjunto de datos de evaluación predichos por el modelo línea de base y el modificado. Se toma MIoU ya que es la métrica de evaluación más directa para evaluar algoritmos de segmentación (ver Sección 3.3.5). D se expresa mediante la Ecuación 4.4.

$$D = MIoU_{UNet} - MIoU_{UNetFFT} \tag{4.4}$$

D se calcula para cada imagen del conjunto de evaluación.

La hipótesis nula (H_0) dice que la mediana de las diferencias entre los valores MIoU de los dos modelos comparados es cero, lo que indica que las diferencias entre los dos modelos no son estadísticamente significativas y se deben al azar.

La hipótesis alternativa (H_a) establece que existe una diferencia significativa en los valores de MIoU de los modelos comparados y no se deben al azar.

Se establece un nivel de significancia $\alpha = 0{,}05$ para una prueba bilateral (de dos colas).

Se calcula el estadístico de Wilcoxon siguiendo los pasos descritos en la Sección 2.11.1. Dado que la cantidad de muestras del conjunto de evaluación es $n = 160$, valor que excede los mostrados en las tablas de significancia de Wilcoxon, se calcula el estadístico Z (el cual sigue una distribución normal) en base al estadístico de Wilcoxon. Z se compara con el valor crítico basado en el nivel de confianza establecido para aceptar o rechazar la hipótesis nula.

4.5.1. Prueba de Wilcoxon: UNet16 y UNetFFT16

En la Tabla 4.4, se muestran los valores medios de MIoU y su desviación estándar en las 10 iteraciones de entrenamiento de los modelos UNet16 y UNetFFT16 sobre el conjunto de datos de evaluación.

Tabla 4.4
Métricas de evaluación de los modelos entrenados.

Modelo	**MIoU (%)**	**Parám. (M)**
UNet16	81.76 ± 2.57	2.16
UNetFFT16	85.40 ± 1.12	8.4

Se evidencia que la media del MIoU alcanzado por el modelo propuesto (UNetFFT16) es superior en 3.64 % al modelo línea de base (UNet16). Para evaluar si esta diferencia es estadísticamente significativa, se toma el mejor entrenamiento de ambos modelos, se realizan las predicciones del conjunto de evaluación (160 imágenes) y se evalúan la significancia estadística mediante la prueba de Wilcoxon para muestras pareadas. Se dice que las muestras son pareadas porque una misma imagen es procesada por dos modelos distintos y se quiere evaluar si la diferencia entre las mediciones de MIoU son estadísticamente significativas o se deben al azar.

Se realiza el cálculo de las diferencias, D, siguiendo la Ecuación 4.4 y se sigue con el cálculo del estadístico de Wilcoxon usando herramientas de software que implementan los procedimientos indicados en la Sección 2.11.1 (para este estudio, se utiliza la librería de Python, Scipy), cuyo código se muestra en el Apéndice H.

El estadístico de Wilcoxon calculado para las 160 muestras evaluadas en los dos modelos es $T = 896$. Normalmente, para evaluar si se acepta o rechaza la hipótesis nula se utiliza la tabla de Wilcoxon, sin embargo, esta tabla no está calculada para muestras grandes (n >50). En estos casos, en base al estadístico de Wilcoxon se calcula el estadístico Z.

La hipótesis nula de la prueba de Wilcoxon no asume una distribución normal de la diferencia entre las mediciones (en este caso MIoU), pero sí una distribución simétrica. Esto es, si la mediana de las diferencias es cero ($mediana(D) = 0$), se asume que su distribución es simétrica y, para muestras grandes (n >50), la distribución de los rangos derivados del ordenamiento de las diferencias (D) tiene una distribución normal debido al teorema central del límite (King

y Eckersley, 2019), por lo que se puede calcular el estadístico Z mediante la Ecuación 4.5.

$$Z = \frac{T - \mu_T}{\sigma_T} \tag{4.5}$$

Donde μ_T representa la media de distribución normal de los rangos del estadístico de Wilcoxon, determinado mediante la Ecuación 4.6.

$$\mu_T = \frac{n(n+1)}{4} \tag{4.6}$$

Y σ_T es la desviación estándar de la distribución normal de los rangos, calculado mediante la Ecuación 4.7.

$$\sigma_T = \sqrt{\frac{n(n+1)(2n+1)}{24}} \tag{4.7}$$

Siendo n el número de muestras evaluadas. Para el caso de estudio, $n = 160$, por lo que reemplazando en las Ecuaciones 4.6 y 4.7, se obtiene:

$$\mu_T = \frac{160(161)}{4} = 6440 \tag{4.8}$$

$$\sigma_T = \sqrt{\frac{160(161)(321)}{24}} = 586{,}97 \tag{4.9}$$

Reemplazando en le Ecuación 4.5, siendo $T = 896$.

$$Z = \frac{896 - 6440}{586{,}97} = -9{,}445 \tag{4.10}$$

Dado que se asume que Z tiene una distribución normal, y dado el nivel de significancia establecido en $\alpha = 0{,}05$ para una prueba de dos colas, se determina el valor crítico para $\frac{\alpha}{2} = 0{,}025$, el cual es 1,96 según la Tabla 4.5 que representan los valores críticos para diferentes niveles de significancia.

Dado que es una prueba de dos colas, el valor crítico asociado es ±1,96. Si el estadístico Z calculado se encuentra en el rango $[-1{,}96, 1{,}96]$, se acepta la hipótesis nula. Si se encuentra fuera del rango anterior, se rechaza la hipótesis nula. Dado que $Z = -9{,}445$, este se encuentra

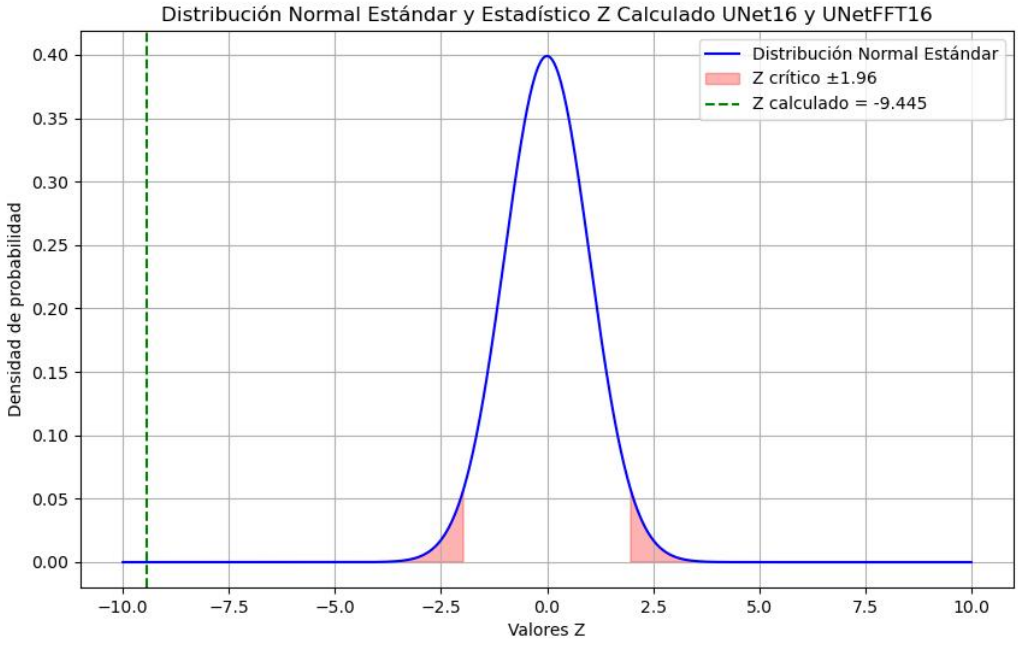
Tabla 4.5
Tabla de valores críticos de la distribución normal estándar.

Z	0.00	0.01	0.02	0.03	0.04	0.05	**0.06**
1.8	0.0359	0.0351	0.0344	0.0336	0.0329	0.0322	0.0314
1.9	0.0287	0.0281	0.0274	0.0268	0.0262	0.0256	**0.0250**
2.0	0.0228	0.0222	0.0217	0.0212	0.0207	0.0202	0.0197

Nota: Tabla modificada de (UNP, 2024).

fuera del rango indicado, por lo que se rechaza la hipótesis nula. Gráficamente se observa esta evaluación en la Figura 4.8.

Figura 4.8
Ubicación del estadístico Z en la distribución normal.



Se observa en la Figura 4.8 la distribución normal junto con los valores críticos de ±1,96 que representan el área (color rojo) para un nivel de significancia $\alpha = 0{,}05$ de una prueba de dos colas. Adicionalmente, se ubica la posición del estadístico Z calculado y se observa que cae fuera del rango $[-1{,}96, 1{,}96]$, por lo que se concluye que se rechaza la hipótesis nula, esto es, la diferencia de las mediciones de MIoU del conjunto de datos de evaluación es estadísticamente significativa y no se debe al azar. Concatenando con la información de la Tabla 4.4, la mejora en el rendimiento debido a la modificación realizada es estadísticamente significativa.

4.5.2. Prueba de Wilcoxon: UNet32 y UNetFFT32

Similarmente a la Sección 4.5.1, se procede con la comparación entre UNet32 y UNetFFT32 para evaluar el impacto de la adición del módulo **Fourier Combination** en el modelo línea de base. En la Tabla 4.6, se muestran los valores de medios MIoU y su desviación estándar en las 10 iteraciones de entrenamiento de los modelos UNet32 y UNetFFT32 sobre el conjunto de datos de evaluación.

Tabla 4.6
Métricas de evaluación de los modelos entrenados.

Modelo	**MIoU (%)**	**Parám. (M)**
UNet32	82.44 ± 3.36	8.64
UNetFFT32	85.94 ± 1.38	33.6

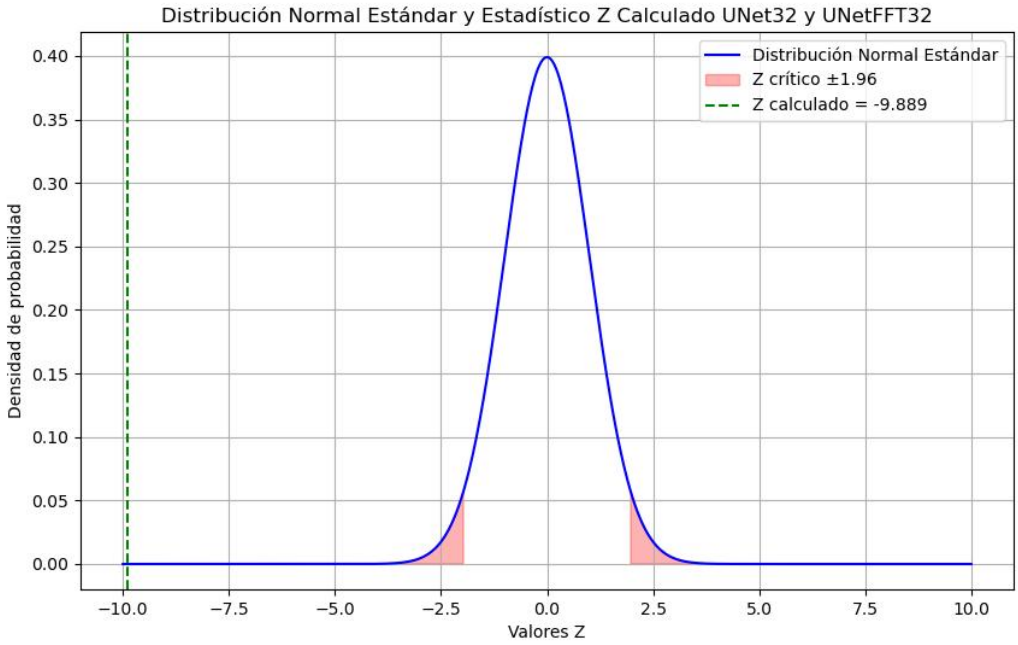
De la Tabla 4.6, se observa que el modelo modificado ofrece un mejor rendimiento general (un 3.5 % superior en la media). Se aplica la prueba de Wilcoxon en las predicciones de las 160 imágenes del conjunto de datos de evaluación, cuyo valor es de $T = 635$. Dado que la μ_T y σ_T dependen de la cantidad de elementos ($n = 160$), los valores de las Ecuaciones 4.8 y 4.9 se mantienen. El estadístico Z para esta comparación viene dado por:

$$Z = \frac{635 - 6440}{586{,}97} = -9{,}889 \tag{4.11}$$

Para el mismo nivel de significancia $\alpha = 0{,}05$ y una prueba de dos colas, los valores críticos son ±1,96. En la Figura 4.9, se observa la ubicación de los valores críticos y el estadístico Z calculado.

Dado que el estadístico $Z = -9{,}889$ se ubica fuera del rango $[-1{,}96, 1{,}96]$, entonces, se rechaza la hipótesis nula, lo que indica que la diferencia de las mediciones de MIoU del conjunto de datos de evaluación es estadísticamente significativa. De acuerdo a la Tabla 4.6, la mejora en el rendimiento debido a la adición del módulo **Fourier Combination** del modelo UNetFFT32 es estadísticamente significativa y no se debe al azar.

Figura 4.9
Ubicación del estadístico Z en la distribución normal.



4.6. Evaluación de la Exactitud del Modelo

En la zona de estudio, se han realizado algunas investigaciones por el Instituto Nacional de Investigación de Glaciares y Ecosistemas de Montaña (INAIGEM). En el Reporte de Peligros en Glaciares RPG 001-2020, Lagunas en formación, cordillera del Vilcanota (INAIGEM, 2020c), se analiza la variación respecto al área superficial de diferentes lagunas mediante la utilización de imágenes satelitales de alta resolución (10m por píxel para las bandas RGB) provenientes del satélite Sentinel 2 L2A. La variación de área se evalúa en los años 2019 y 2020, tomándose las imágenes disponibles y sin nubosidad de las fechas indicadas en la Tabla 4.7.

Tabla 4.7
Fechas de adquisición de imágenes de estudio de referencia de INAIGEM.

Satélite	**Fecha**
Sentinel 2 L2A	17 de julio 2019 (17/07/2019)
Sentinel 2 L2A	27 de agosto 2020 (27/08/2020)

Nota: Tabla obtenida de (INAIGEM, 2020c).

El estudio en mención calculó las áreas en estas dos fechas y determinó las variaciones

porcentuales por año, concluyendo que algunas de ellas presentaban alto riesgo de desborde por derretimiento glaciar, considerando realizar estudios más detallados. Las lagunas estudiadas se separaron por sectores dependiendo a qué nevado pertenecían, se consideran los siguientes grupos: a) Ausangate, b) Anante, c) Huicachani, d) Yanajasa y e) Chumpe; a cada laguna se le da una denominación considerando al grupo al que pertenece y un número asociado, estas denominaciones, sus ubicaciones UTM y las áreas determinadas se resumen en la Tabla 4.8.

Tabla 4.8
Denominación, ubicación UTM y área de lagunas en estudio de referencia.

Grupo	**Código**	**UTM E** (m)	**UTM S** (m)	**Área 2019** (m^2)	**Área 2020** (m^2)
Ausangate	A-10	256262	8475969	102 933	178 019
Ausangate	A-20	259137	8477247	4 968	8 948
Ausangate	A-30	262543	8475575	71 121	84 706
Ausangate	A-40	260990	8472500	182 582	218 984
Anante	AN-10	263118	8481165	253 201	275 542
Anante	AN-20	265031	8477393	12 579	22 200
Anante	AN-30	264084	8477472	8 008	4 858
Huiscachani	H-10	271298	8482796	96 086	146 976
Huiscachani	H-20	270063	8484363	88 276	94 797
Huiscachani	H-30	278838	8482812	53 445	68 761
Yanajasa	Y-10	296055	8479392	125 255	134 133
Chumpe	CH-10	276093	8476534	38 424	39 156

Nota: Tabla modificada de (INAIGEM, 2020c).

De acuerdo a (INAIGEM, 2023), que consideró $5000m^2$ como el área mínima cartografiable, en este análisis de exactitud del modelo se omiten las lagunas con dimensiones menores a $5000m^2$, esto es, no se consideran las lagunas con denominaciones A-20 y AN-30.

Para comprobar la exactitud en la estimación de área del modelo propuesto, se toma el mejor entrenamiento de la red UNetFFT32 que obtuvo las mejores métricas y se compara su estimación de área respecto a las mediciones de referencia realizadas por INAIGEM, que además usaron imágenes de mayor resolución espacial. Se calcula el error absoluto y relativo en la estimación de área para cada laguna y se evalúa si los resultados son equivalentes desde un punto estadístico mediante la prueba de Wilcoxon para muestras pareadas.

Para realizar las predicciones se obtienen imágenes satelitales provenientes del satélite Landsat-8 que sean de fechas aproximadas a las tomadas por el estudio de referencia y que

contengan poca cantidad de nubosidad. Las escenas obtenidas se detallan en la Tabla 4.9, todas tienen el mismo nivel de procesamiento que las imágenes usadas durante el entrenamiento (L2SP).

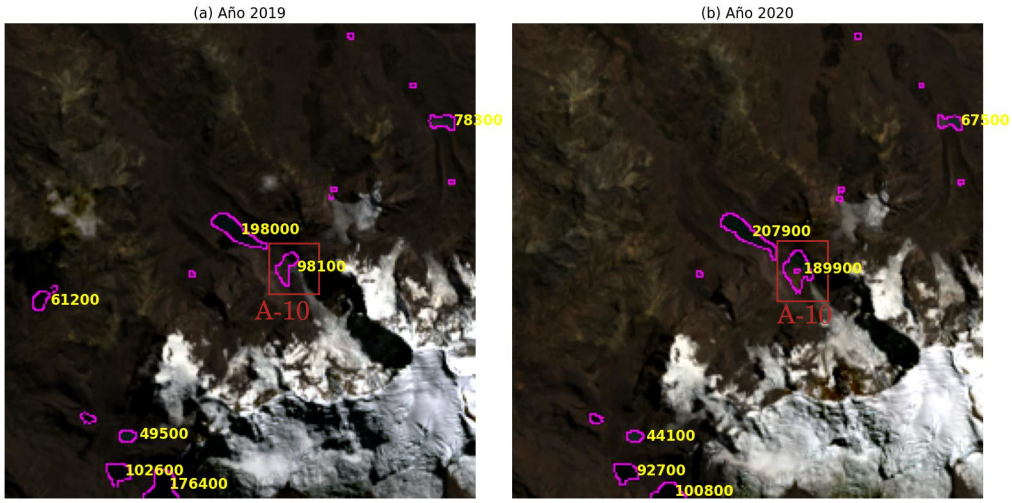
Tabla 4.9
Fechas de adquisición de imágenes de estudio de referencia de INAIGEM.

Satélite	**Fecha de captura**
Landsat-8	07 de julio 2019 (07/07/2019)
Landsat-8	26 de agosto 2020 (26/08/2020)

Se observa de la Tabla 4.9 que la imagen de Landsat para 2019 fue obtenida 10 días antes de la referencia, mientras que la de 2020, apenas 1 día antes. Con estos elementos, se procede a realizar la predicciones de las zonas de interés (lagunas de referencia).

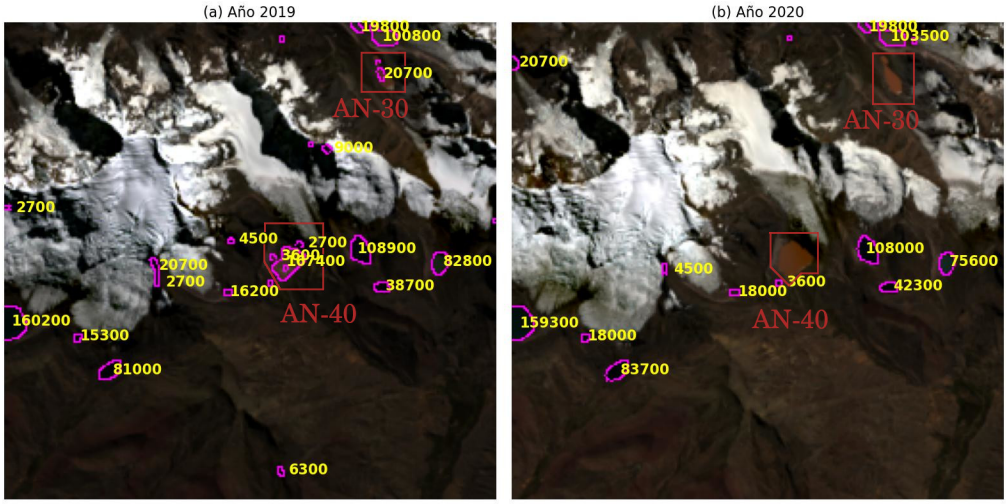
En las Figuras 4.10, 4.11, 4.12, 4.13, 4.14, 4.15 y 4.16 se observan las predicciones realizadas por el modelo UNetFFT32 superpuestas en la representación RGB de las lagunas estudiadas. Además, se muestra en color amarillo la estimación de área de los cuerpos de agua detectados. Para el cálculo de la estimación de área se toma en cuenta la resolución del píxel, que según la Tabla 3.1 para Landsat-8 es de $30m$, por lo que cada píxel identificado como laguna, representa un área de $900m^2$. El cálculo de la estimación de área del modelo se realiza multiplicando la cantidad de píxeles clasificados como lagunas que estén agrupados por el área del píxel ($900m^2$).

Figura 4.10
Predicción de laguna A-10 con modelo UNetFFT32.



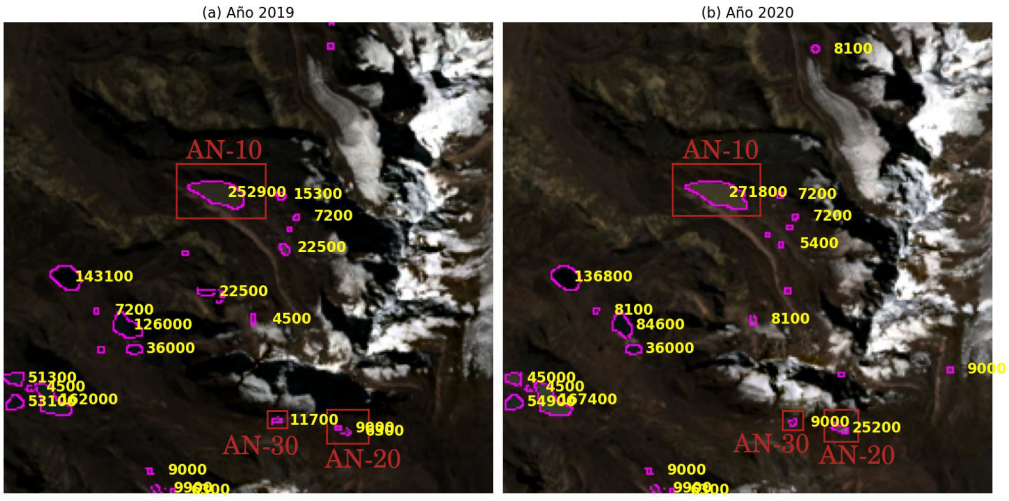
Se observa en la Figura 4.11 una situación especial, en el año 2019 la laguna A-30, es apenas detectada por el modelo UNetFFT32, mientras que en el año 2020, A-30 es totalmente ignorada.

Figura 4.11
Predicción de laguna A-30 y A-40 con modelo UNetFFT32.



Por otro lado, respecto a la laguna A-40, en el año 2019 esta es detectada por el modelo, mientras que en el año 2020 esta es ignorada. Esta situación es desarrollada en la Sección 4.7. Para el propósito de evaluar la exactitud del modelo, entonces, se descartan estas lagunas ignoradas por el modelo, ya que no es posible realizar una comparación numérica entre los valores de referencia y predicciones, estableciendo una limitación al modelo propuesto.

Figura 4.12
Predicción de laguna AN-10, AN-20 y AN-30 con modelo UNetFFT32.



Con estas consideraciones de la Figura 4.11, en la Tabla 4.10, se condensan las mediciones de área de referencia y la estimación de área establecida por el modelo UNetFFT32, así como el error absoluto (Ecuación 4.12) y error relativo (Ecuación 4.13).

Figura 4.13
Predicción de laguna H-10 y H-20 con modelo UNetFFT32.

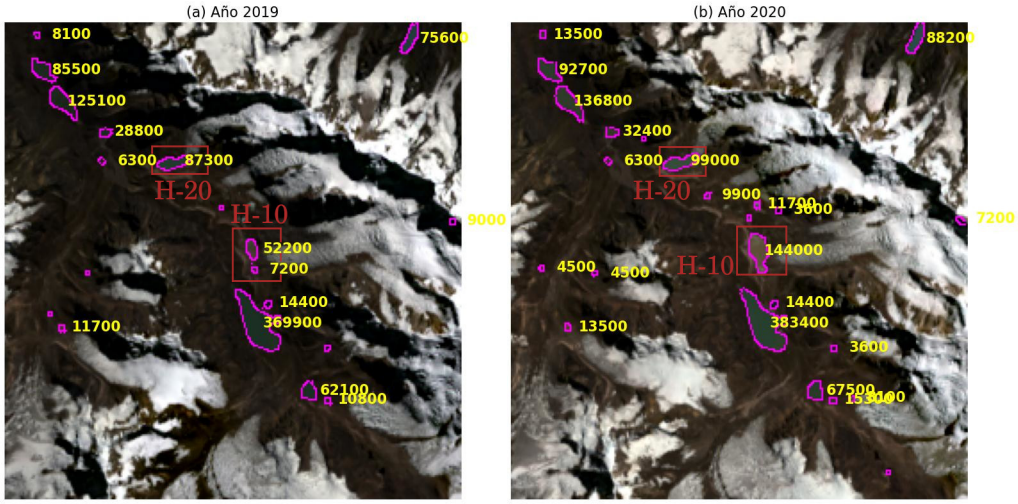
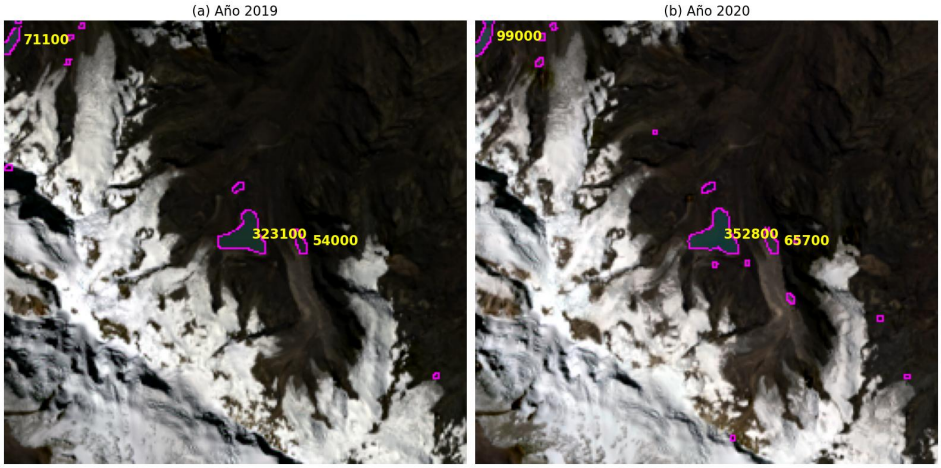


Figura 4.14
Predicción de laguna H-30 con modelo UNetFFT32.



$$E_a = |\text{Valor observado} - \text{Valor real}| \tag{4.12}$$

$$E_r = \frac{E_a}{\text{Valor real}} \times 100\,\% \tag{4.13}$$

Donde valor observado representa las estimaciones de área predichas por el modelo UNetFFT32 y valor real representa los valores de referencia establecidos por (INAIGEM, 2020c).

De la Tabla 4.10, se observa que el error relativo promedio es de 6.33 %. Además se observan 3 datos con altos valores de error relativo: a) AN-20-2019, b) AN-20-2020 y c) H-10-2019.

Figura 4.15
Predicción de laguna Y-10 con modelo UNetFFT32.

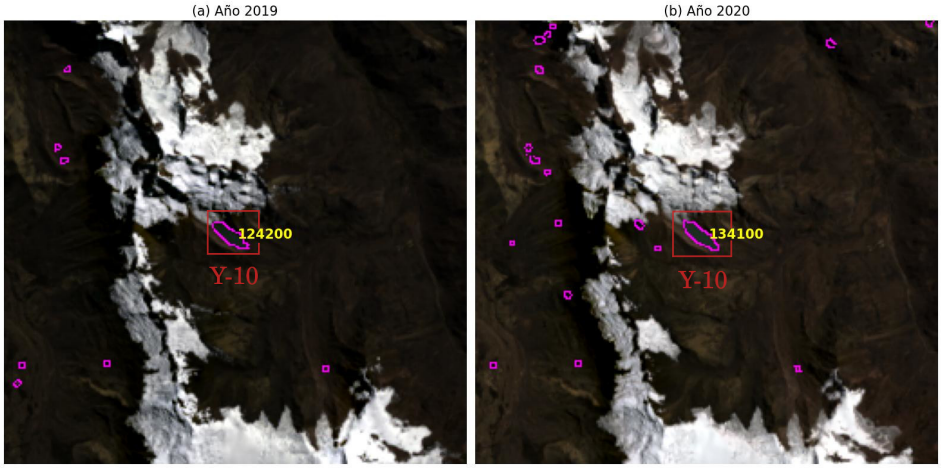
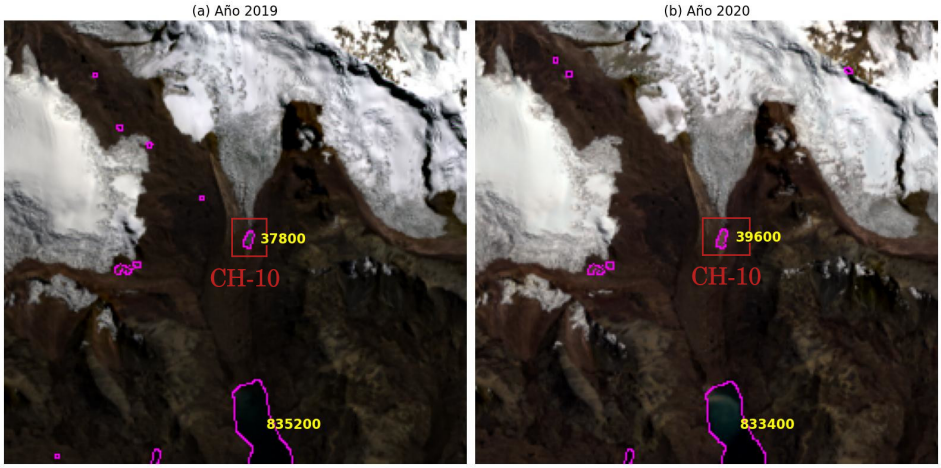


Figura 4.16
Predicción de laguna CH-10 con modelo UNetFFT32.

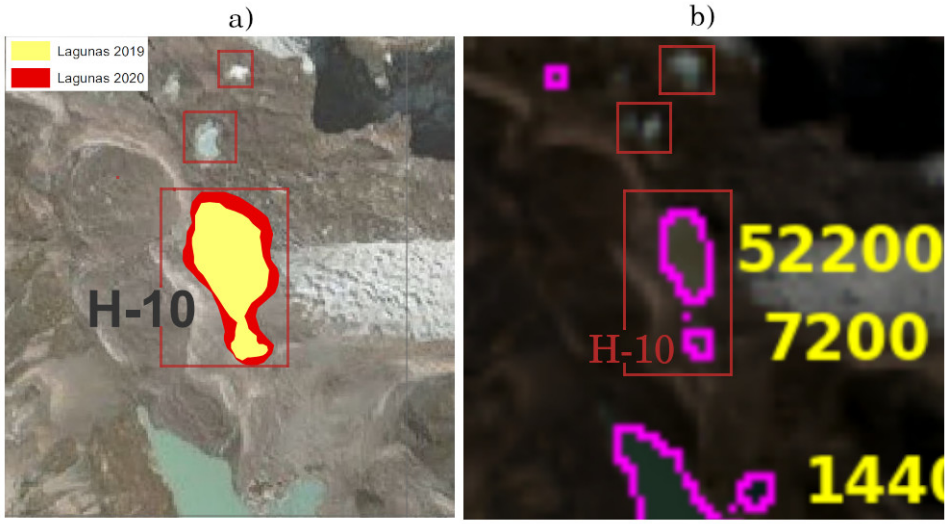


Respecto a la laguna AN-20, en el año 2019, se observa un error relativo del 21.63 % (con 2 721 m de error absoluto, representando alrededor de 3 píxeles). Se observa que aunque el error porcentual es elevado, el error absoluto representa apenas 3 píxeles, lo que indica que para lagunas pequeñas (AN-20, 12579m^2) pequeños errores absolutos pueden representar altos valores de error relativo. Sucede similarmente con A-20 en el año 2020, con un error del 13.51 % (3000 m de error absoluto, representando cerca de 3.3 píxeles). Sobre el último caso, existe un gran error relativo y absoluto respecto a la laguna H-10 el año 2019 (38.18 % y 36 686 m, alrededor de 40 píxeles), la comparación entre la segmentación y estimación de área del modelo UNetFFT32 junto con el área identificada como laguna por la investigación de referencia se muestra en la Figura 4.17.

Tabla 4.10
Error absoluto y relativo de las estimaciones de área realizadas por UNetFFT32.

Laguna	Año	Área Referencia (m^2)	Área UNetFFT32 (m^2)	Error Absoluto (m^2)	Error Relativo
A-10	2019	102 933	98 100	4 833	4.69 %
A-10	2020	178 019	189 900	11 881	6.67 %
A-40	2019	182 582	173 700	8 882	4.86 %
AN-10	2019	253 201	252 900	301	0.12 %
AN-10	2020	275 542	271 800	3 742	1.35 %
AN-20	2019	12 579	15 300	2 721	21.63 %
AN-20	2020	22 200	25 200	3 000	13.51 %
H-10	2019	96 086	59 400	36 686	38.18 %
H-10	2020	146 976	144 000	2 976	2.02 %
H-20	2019	88 276	87 300	976	1.10 %
H-20	2020	94 797	99 000	4 203	4.43 %
H-30	2019	53 445	54 000	555	1.04 %
H-30	2020	68 761	65 700	3 061	4.45 %
Y-10	2019	125 255	124 200	1 055	0.84 %
Y-10	2020	134 133	134 100	33	0.02 %
CH-10	2019	38 424	37 800	624	1.62 %
CH-10	2020	39 156	39 600	444	1.13 %

Figura 4.17
Comparación entre predicción por UNetFFT32 y área de referencia de laguna H-10.



Se observa una discrepancia entre la segmentación de referencia (Figura 4.17 a)) que en color amarillo describe el área para el año 2019 (12579m^2) y la predicción del modelo (Figura 4.17 b)) que en color morado rodea la laguna segmentada. Respecto a esta última, no se observa una

continuidad en la segmentación, así también, existe una diferencia significativa respecto a los pequeños cuerpos de agua en la parte superior. Estas diferencias pueden explicarse debido a las diferentes fechas de adquisición de las imágenes, posiblemente estas lagunas son superficiales y de reciente formación, por lo que cambios recientes pueden tener grandes efectos en su área, tanto en la laguna de estudio como en las lagunas aledañas (resaltadas en cuadros de color rojo).

Para evaluar la exactitud de las predicciones y estimaciones de área del modelo propuesto respecto a las informaciones de área de referencia, se aplica la prueba de Wilcoxon para muestras pareadas, debido a que se evalúa el área de la misma laguna por dos métodos diferentes, además, debido a la cantidad limitada de datos, no se puede asumir una distribución normal de las muestras. La hipótesis nula y alternativa se muestran en las Ecuaciones 4.14 y 4.15.

$$H_0 : mediana(D_A) = 0 \tag{4.14}$$

$$H_a : mediana(D_A) \neq 0 \tag{4.15}$$

Donde D_A representa las diferencias de los valores de área de referencia y estimado por el modelo para cada muestra, que se expresa mediante la Ecuación 4.16.

$$D_A = A_r - A_{UNetFFT32} \tag{4.16}$$

Siendo A_r el área de la muestra establecida por el informe de referencia (INAIGEM, 2020c) y $A_{UNetFFT32}$ es la estimación de área de la misma muestra.

Respecto a la Ecuación 4.14 que describe la hipótesis nula, esta indica que no hay diferencia en las áreas medidas por el informe de referencia y el modelo de UNetFFT32. Esto significa que las diferencias entre las dos mediciones son, en promedio, cero.

Por otro lado, la Ecuación 4.15 describe la hipótesis alternativa e indica que existe una diferencia en las áreas medidas por el informe de referencia y el modelo de UNetFFT32. Esto significa que, en promedio, una de las mediciones tiende a ser mayor o menor que la otra.

El cálculo del estadístico de Wilcoxon se realiza utilizando la librería Scipy de Python que implementa los procesos necesarios para su cálculo, basado en las diferencias entre las muestras

pareadas. Para una distribución de dos colas con significancia de $\alpha = 0{,}05$, el estadístico es $T = 54$. Ya que la cantidad de muestras es $n = 17$, se puede aceptar o rechazar la hipótesis nula basados en la Tabla 4.11 que indica los valores críticos ya calculados para la prueba de Wilcoxon para diferentes cantidades de muestras.

Tabla 4.11
Valores críticos para el estadístico de Wilcoxon para diferentes cantidades de muestras pareadas y niveles de significancia para prueba de dos colas.

n	$\alpha = 0.01$	α **= 0.05**	$\alpha = 0.1$
16	19	29	35
17	23	**34**	41
18	27	40	47

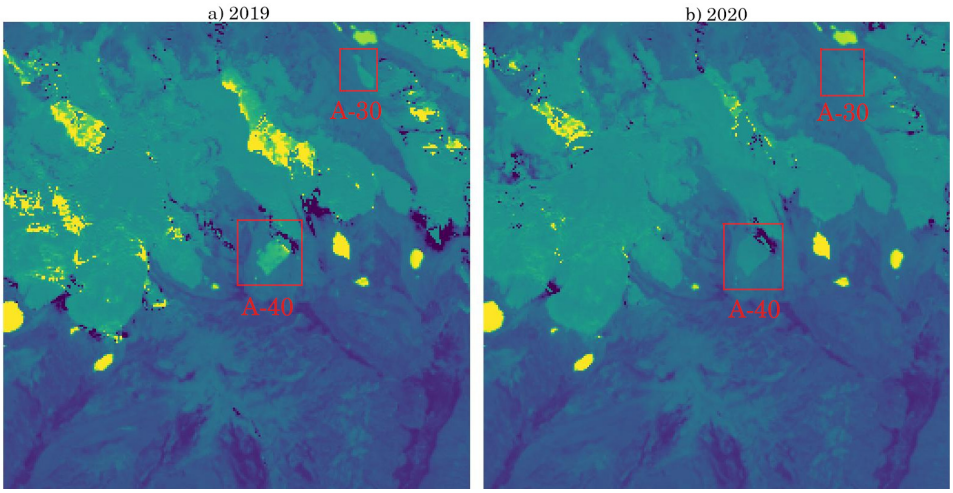
Si el valor del estadístico de Wilcoxon es mayor al valor crítico, no se puede rechazar la hipótesis nula. Dado que el valor del estadístico $T = 54 > V_c = 34$, T es superior al valor crítico calculado para $n = 17$ muestras, entonces, no hay suficiente evidencia para rechazar la hipótesis nula. Por lo tanto, para un nivel de significancia de $\alpha = 0{,}05$, se concluye que no existen diferencias estadísticamente significativas entre las dos mediciones de áreas para las lagunas estudiadas.

4.7. Limitaciones del Modelo

En la Figura 4.11, se observó que el modelo UNetFFT32 ignora A-30 y A-40. Se observa que estas lagunas tienen un color predominantemente marrón que representa su turbidez, esto se debe a que estas lagunas de formación reciente tienen partículas en suspensión principalmente de arcilla (INAIGEM, 2020a). Esta composición afecta en las propiedades espectrales del agua, esto se aprecia en la Figura 4.18, que muestra la representación NDWI de la laguna A-30 y A-40 para el año 2019 y 2020.

Para el año 2019, se observa que tanto A-30 como A-40 su representación NDWI es bastante tenue, lo que indica que sus propiedades reflectantes de las bandas verde e infrarrojo cercano han cambiado en gran medida. Para el año 2020 se observa que estas dos lagunas han perdido prácticamente sus características espectrales, por lo que casi se confunden con el fondo (nieve y suelo).

Figura 4.18
Representación NDWI de las lagunas A-30 y A-40 en el año 2019 y 2020.



Los modelos de aprendizaje generalizan sus tareas en base a las muestras entregadas en el entrenamiento. Dado que estas lagunas de alta turbidez son bastante limitadas en la zona de estudio, el modelo no ha podido generalizar la segmentación de estos cuerpos de agua, cuyas características espectrales son bastante diferentes de otros cuerpos de agua más regulares, por lo que una limitación del modelo es la segmentación de lagunas con alta turbidez.

4.8. Cuadro Resumen de Resultados

En la Tabla 4.12, se resumen los hallazgos más significativos obtenidos de los experimentos realizados, en los que se compara el rendimiento del modelo propuesto, UNetFFT, con los otros métodos de comparación. Se tiene una columna para la metodología de comparación NDWI con umbralización por Otsu así como para cada modelo de comparación utilizado en la investigación. Las dos últimas columnas representan los valores alcanzados por cada una de las instancias del modelo propuesto, UNetFFT16 y UNetFFT32. Resaltados en negrita se muestran los mejores resultados para cada criterio expuesto, como son, métricas MIoU, F1-Score y *Pixel Accuracy* (PA) promedio, las épocas de entrenamiento requeridas, la cantidad de parámetros. Cualitativamente se evalúa la robustez del modelo, considerada como la capacidad de los modelos de diferenciar los lagos y lagunas de su fondo, pudiendo diferenciarlos entre sombras, nubes y nieve. También se evalúan las capacidades de detección de bordes y lagunas pequeñas. Se indica

el valor del estadístico Z de Wilcoxon para muestras pareadas que representan la significancia estadística en la mejora del rendimiento del modelo propuesto respecto a las instancias del modelo línea de base. Finalmente, se indica el error porcentual obtenido en el experimento de exactitud del mejor modelo, UNetFFT32, respecto al estudio de referencia (INAIGEM, 2020c).

Se observa de los datos de la Tabla 4.12 que tanto UNetFFT16 como UNetFFT32, instancias del modelo propuesto, obtienen los mejores resultados y solucionan los problemas de detección pobre y borrosa de bordes, obteniendo las mejores métricas y siendo capaces de realizar segmentaciones más precisas en los bordes y en lagunas pequeñas. Específicamente, el modelo UNetFFT32 obtuvo el mejor rendimiento a todos los otros modelos propuestos, contando con 33.6 millones de parámetros, siendo uno de los más altos. También se observa que estas mejoras introducidas por el modelo propuesto son estadísticamente significativas mediante la prueba de Wilcoxon. Dados estos resultados, se escoge este modelo, UNetFFT32, para la realización de las estimaciones de área y medición de exactitud. Finalmente, se observa que el error porcentual en la estimación de áreas por el mejor modelo (UNetFFT32) es de 6.33 %.

Tabla 4.12

Cuadro resumen de resultados de los experimentos.

Criterios	NDWI	FCN32	FCN64	Linknet32	Linknet64	PSPNet32	PSPNet64	UNet16	UNet32	UNet64	UNet FFT16	UNet FFT32
MIoU promedio (%)	46.33	82.70	83.37	82.07	82.49	80.87	80.96	81.76	82.44	79.27	85.40	**85.94**
F1-S promedio (%)	26.45	76.70	78.44	75.51	76.21	73.39	73.79	74.35	75.38	69.29	80.69	**81.60**
PA promedio (%)	70.41	99.59	99.63	99.57	99.58	99.53	99.53	99.60	99.62	99.51	**99.67**	**99.67**
Épocas para su mejor valor	-	42	52	47	38	**31**	62	39	47	47	130	175
Cantidad de Parámetros	-	7.57M	30.26	2.84M	11.30M	11.77M	46.94M	**2.16M**	8.64M	34.53M	8.4M	33.6M
Robusto	No	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**
Detección de bordes	**Preciso**	Impreciso	Regular	Regular	Regular	Impreciso	Impreciso	Regular	Regular	Regular	**Preciso**	**Preciso**
Detección lagunas pequeñas	**Sí**	No	No	No	No	No	No	No	No	No	**Sí**	**Sí**
Prueba de Wilcoxon (Z) (Ablación)	-	-	-	-	-	-	-	■	★	-	-9.445 (Respecto a UNet16, ■)	-9.889 (Respecto a UNet32, ★)
Error porcentual promedio estimación de área	-	-	-	-	-	-	-	-	-	-	-	6.33 %

Capítulo 5

Discusión de Resultados

En esta sección se discuten los resultados obtenidos, identificando los aportes del modelo propuesto, sus ventajas y limitaciones, así como su contextualización respecto a los objetivos planteados en la presente tesis.

La presente investigación propone la implementación de un modelo basado en el paradigma del aprendizaje profundo para realizar la tarea de segmentación de lagunas en la cordillera del Vilcanota. El modelo propuesto busca resolver algunas limitaciones identificadas en la literatura como baja precisión en la segmentación y detección de bordes borrosos de los cuerpos de agua. Para esta investigación se toma como línea de base al modelo UNet, una arquitectura codificador-decodificador ampliamente utilizada y con excelentes resultados en el campo de la segmentación de imágenes. La modificación realizada al modelo línea de base consiste en la aplicación de filtros en el dominio de la frecuencia, específicamente en las conexiones de salto. El rendimiento del modelo se compara con otras arquitecturas utilizadas en la tarea de segmentación mediante métricas estándar para esta tarea y su exactitud es evaluada con mediciones de área realizadas por INAIGEM, una institución gubernamental peruana encargada de realizar estudios en ecosistemas de alta montaña.

Los modelos de comparación (FCN, Linknet, PSPNet), así como el modelo línea de base (UNet) basan su funcionamiento en la aplicación de CNN, las cuales aprovechan su capacidad de recopilar información global y detectar características a diversas escalas, específicamente mediante la aplicación de capas de agrupamiento (*pooling*) y conexiones residuales, sin embar-

go, a pesar de ello, pasan por alto relaciones espaciales entre diferentes objetos en la imagen (Amyar et al., 2022). Para realizar actividades de monitoreo y seguimiento de lagunas basadas en imágenes satelitales, hacen falta algoritmos que sean rápidos y precisos en la segmentación y estimación de área de los cuerpos de agua estudiados; en este contexto, el desarrollo del modelo propuesto busca mejorar las segmentaciones realizadas, para ello, toma como modelo línea de base a UNet, una red que para tareas de segmentación en general ha obtenido excelentes resultados y que no requiere de extensos datos de entrenamiento. Para abordar los problemas detectados en este tipo de redes se propone la aplicación de filtros en el dominio de la frecuencia, los cuales son especialmente efectivos al trabajar sobre imágenes con presencia de ruido, además, mediante filtros pasa bajos pueden desenfocar la imagen, mientras que mediante filtros pasa altos pueden realzar detalles finos de forma controlada (Gonzales y Woods, 2008). El modelo propuesto, llamado UNetFFT, complementa la extracción de características espaciales (mediante CNN) con la combinación de estas características en el dominio de la frecuencia, usando filtros aprendibles que procesan los mapas de características detectando detalles finos en las imágenes pudiendo segmentar pequeñas lagunas y mejorando la detección fina de bordes.

En la comparación de resultados, se observa que la metodología que utiliza el índice NDWI junto con el algoritmo de umbralización Otsu para la segmentación de lagunas, obtuvo los peores resultados, debido principalmente a que la zona de estudio tiene una variabilidad geográfica y climática compleja, con presencia de nieve y cordilleras que generan sombras, ambas con comportamiento espectral similar al agua, elementos que perjudican la segmentación precisa de lagunas con esta metodología. Por otro lado, las instancias de los modelos de comparación (FCN, Linknet, PSPNet y UNet) obtuvieron en promedio un 81 % para la métrica MIoU, 74 % para F1-Score y 99.57 % para PA. En general, estos modelos de comparación han obtenido buenos resultados en las métricas medidas, siendo el mejor de ellos FCN32, obteniendo un MIoU de 84.37 % con 30.26 millones de parámetros. Por otro lado, el modelo de comparación de menor tamaño (2.16 millones de parámetros) es UNet16 y obtuvo un 81.76 % de MIoU promedio, siendo un resultado bastante competitivo, lo que revela la capacidad del modelo UNet para capturar y generalizar incluso con pocas muestras.

Respecto a las instancias del modelo propuesto, UNetFFT16 y UNetFFT32, ambos modelos

obtuvieron los mejores resultados para todas las métricas, superando a todos los modelos de comparación. En específico, UNetFFT32 obtuvo en promedio un MIoU de 85.94 %, lo que refleja una superposición adecuada entre lo predicho por el modelo y las máscaras de verdad; obtuvo también el F1-Score más alto de 81.60 % en promedio, demostrando que el modelo diferencia adecuadamente los elementos que son considerados lagos como los que no, minimizando tanto los falsos positivos como falsos negativos. Finalmente, se observa que UNetFFT16 obtuvo el mejor valor para PA (99.67 %) junto a UNetFFT32. Es de notar que, en general, para los modelos de comparación los valores de PA son bastante altos, esto se debe a las características de las máscaras conjunto de datos, como se observó en la Sección 3.1.6 en la Figura 3.14, existe un desbalance de clases, siendo la clase no-laguna la predominante, entonces, incluso aunque los modelos predijeran siempre como si no hubieran lagos en ningún parche, esta métrica sería elevada, ya que PA, según la Ecuación 3.39, depende directamente de los valores de TP y TN, siendo TN los píxeles de la clase dominante no-laguna.

La instancia UNetFFT16 es el segundo mejor modelo de comparación, con apenas 8.4 millones de parámetros, superando a otros modelos con similar cantidad de parámetros (FCN32, Linknet64, PSPNet32 y UNet32). La instancia UNetFFT32 con 33.6 millones de parámetros superó a todos los modelos, incluso a otros con similar y mayor cantidad de parámetros (FCN64, PSPNet64 y UNet64). En un contexto de recursos computacionales limitados, el modelo UNetFFT16 podría ser más adecuado, mientras que si se desea el mejor resultado, UNetFFT32 sería la mejor elección.

Otro aspecto a considerar es que el modelo propuesto (UNetFFT) es más estable en el proceso de entrenamiento, siendo menos sensible parámetros aleatorios, aunque ciertamente tiende a converger más lentamente a comparación de los otros modelos, esto puede deberse a que el modelo hace variaciones más finas en los filtros aprendibles en la frecuencia, aunque con resultados más satisfactorios.

Del análisis cualitativo de los modelos, se observa que todos diferencian adecuadamente las lagunas de su fondo complejo, producto de las cordilleras, sombras y nieve. Sin embargo, el modelo propuesto, UNetFFT, supera a todos los demás en la detección fina de bordes, delineando mejor los cuerpos de agua, lo que se traduce en mejores segmentaciones, así

también, el modelo propuesto es capaz de detectar pequeños cuerpos de agua, aunque con dificultades para segmentar lagunas superficiales con presencia de vegetación. Esta limitación puede deberse a que este tipo de lagunas es poco común en el contexto de la zona de estudio y el modelo no tuvo suficientes muestras de este tipo para generalizar.

En el estudio de ablación, se muestra que la adición del módulo **Fourier Combination** en el modelo línea de base UNet mejora los resultados de segmentación, tanto numéricamente, mediante las métricas, como cualitativamente, observando la detección de bordes finos, lo cual es un efecto de la adición de los filtros en la frecuencia que pueden resaltar de mejor manera los bordes presentes en las imágenes y combinando de mejor manera los datos del codificador y decodificador. Para corroborar que esta mejora es estadísticamente significativa, se aplica el estadístico de Wilcoxon para muestras pareadas, resultando para los dos modelos instanciados (UNetFFT16 y UNetFFT32) en el rechazo de la hipótesis nula, esto es, que la mejora en la métrica MIoU para los modelos propuestos respecto a los modelos línea de base no se debe al azar y es estadísticamente significativa.

Finalmente, en el experimento de evaluación de exactitud del modelo, se comparan las estimaciones de área realizadas por el modelo propuesto, UNetFFT y se compara con datos de área provenientes de una institución de confianza (INAIGEM). Se calcula el error absoluto y relativo para cada laguna estudiada, observándose que, para la mayoría de lagunas, el error absoluto es de $5000m^2$ aproximadamente, lo que representa alrededor de 5.6 píxeles de error por predicción del modelo propuesto. Para lagunas pequeñas, este error absoluto puede representar grandes variaciones en el error relativo. También se aprecian casos atípicos en la comparación de áreas, las cuales pueden explicarse debido a la diferencia de fechas en la adquisición de imágenes de comparación y las características de las lagunas en cuestión, que son superficiales y de reciente formación. Los resultados de las diferencias entre las áreas se comprueban estadísticamente con la prueba de Wilcoxon para muestras pareadas, no pudiendo rechazarse la hipótesis nula, lo que indica que no hay suficientes diferencias estadísticamente significativas entre las mediciones de áreas para las lagunas estudiadas mediante la referencia y el modelo de segmentación propuesto.

Durante la evaluación de resultados se apreció que el modelo tiene dificultad en la detección de lagunas con alta turbidez, debido a que la presencia de arcilla modifica la respuesta espectral

de estas lagunas, lo que dificulta al modelo en identificarlo como agua, confundiéndolo con nieve o suelo. Esto puede deberse a que, siendo este tipo de lagunas poco comunes en la zona de estudio, existen pocas muestras de este tipo en el conjunto de datos generado, lo que no permite el modelo la generalización adecuada de estas lagunas. Por lo que, lagunas con alta turbidez así como lagunas con presencia de vegetación acuática son las principales limitaciones del modelo propuesto, principalmente por la poca cantidad de muestras de este tipo de cuerpos en el conjunto de datos elaborado.

Conclusiones

1. Se implementó un método basado en el paradigma del aprendizaje profundo para la segmentación automática de lagunas en el contexto geográfico de la cordillera del Vilcanota, región Cusco, Perú. El método de segmentación consistió en el desarrollo de un nuevo modelo de aprendizaje profundo sustentado en primer lugar por las redes neuronales convolucionales (CNN) para la extracción de características espaciales y en segundo lugar por los filtros en el dominio de la frecuencia que combinan y procesan los mapas de características de los diferentes niveles de la red de tipo codificador-decodificador, logrando una adaptación a un entorno tan complejo en términos de clima y geografía como lo es el ecosistema de alta montaña al que pertenece la cordillera del Vilcanota. Los resultados de este modelo no solo fueron competitivos sino superiores a las metodologías y modelos utilizados como comparación, demostrando la efectividad de la arquitectura propuesta.

2. Se generó un conjunto de datos específico de lagunas para la zona de estudio, esto es, la cordillera del Vilcanota. Siguiendo consideraciones de la literatura, el conjunto de datos consiste en 800 imágenes de 256×256 píxeles y de 6 bandas: azul (B), verde (G), rojo (R), infrarrojo cercano (NIR) e infrarrojo de onda corta 1 y 2 (SWIR1 y SWIR2). Las máscaras respectivas fueron generadas manualmente usando como referencia las representaciones RGB, NDWI y con una comprobación cruzada utilizando imágenes de alta resolución de Google Earth. Del análisis del conjunto de datos, el 97.94 % de los píxeles representan la clase no-laguna y el 2.06 % representa la clase laguna, siendo esta última la clase no dominante. La fuente de las imágenes es el satélite Landsat-8 con nivel de procesamiento L2SP, que considera correcciones geométricas y radiométricas. Adicionalmente, se realiza el preprocesamiento de los datos que consiste en la normalización Max-Min y relleno de

valores faltantes, lo que permite a los modelos entrenados generalizar de mejor manera los datos de cada banda espectral.

3. Se implementó y entrenó el modelo propuesto, UNetFFT, una arquitectura que combina el procesamiento en el dominio del espacio, mediante redes convolucionales, y el procesamiento en el dominio de la frecuencia, mediante la aplicación de filtros aprendibles en el dominio de la frecuencia. Para la configuración de los parámetros de entrenamiento se siguieron las consideraciones propuestas en la literatura para la tarea de segmentación de lagunas. Del análisis cualitativo, se observa que el modelo propuesto y entrenado es robusto diferenciando las lagunas de su fondo, esto es, de sombras, cordilleras y nieve; además, es capaz de detectar pequeñas lagunas, aunque tiene dificultades para segmentar adecuadamente lagunas con alta turbidez y con presencia de vegetación acuática superficial. El modelo propuesto también mejora la detección de bordes finos, lo que evidencia el aporte del procesamiento en el dominio de la frecuencia.

4. A nivel cuantitativo, la eficacia del modelo propuesto, UNetFFT, se compara con el método de segmentación tradicional que utiliza el NDWI junto con la umbralización por Otsu así como con otros modelos clásicos de segmentación, tales como FCN, Linknet, PSPNet y UNet. La eficacia se mide mediante las métricas que evalúan la calidad de segmentación, las cuales son MIoU, F1-Score y Pixel Accuracy, donde la instancia del modelo UNetFFT32 obtuvo los mejores resultados con un MIoU promedio de 85.94 %, F1-Score promedio de 81.60 % y Pixel Accuracy promedio de 99.67 %, superando a los modelos de comparación que en promedio obtuvieron 81.80 % para MIoU, 74.80 % para F1-Score y 99.57 % para Pixel Accuracy, superando también al método que utiliza NDWI con Otsu, siendo este último el que obtuvo las métricas más bajas. Los resultados indican que el modelo propuesto, UNetFFT, es superior tanto cuantitativa como cualitativamente frente a otras técnicas y metodologías clásicas, abordando las problemáticas de segmentación precisa y de mejor calidad. Estos resultados fueron evaluados estadísticamente mediante la prueba de Wilcoxon para muestras pareadas que demostró que la mejora introducida por el módulo propuesto Fourier Combination en el modelo línea de base UNet es estadísticamente significativa y no se debe al azar.

5. Se evaluó la exactitud de las predicciones del modelo UNetFFT y su respectiva estimación de área en metros cuadrados (m^2) comparándolo con valores de área obtenidos en un estudio previo de INAIGEM, una institución de confianza reconocida en el área. Se tomaron 9 lagunas con sus respectivas mediciones en el año 2019 y 2020. De la comparación del área de referencia y el estimado por UNetFFT, se obtuvo que el error relativo promedio del modelo propuesto es de 6.33 %. Se comprobó mediante la prueba de Wilcoxon para muestras pareadas que no existe una diferencia estadísticamente significativa entre las mediciones reales de referencia y las estimaciones de área de UNetFFT.

Recomendaciones

- La investigación en aprendizaje profundo para segmentación de cuerpos de agua puede beneficiarse de un preprocesamiento de datos más robusto. Se recomienda explorar y experimentar con diversas técnicas de normalización y estandarización de datos, evaluando cómo estas afectan el rendimiento del modelo. Además, la incorporación de distintas combinaciones de bandas espectrales y el uso de índices como NDWI (Índice de Agua Normalizado) o NDVI (Índice de Vegetación Normalizado) podría mejorar la capacidad del modelo para diferenciar cuerpos de agua de otros elementos del paisaje.

- Para un análisis más preciso de las variaciones de las lagunas, se recomienda el uso de imágenes satelitales con mayor resolución espacial y temporal. Esto permitiría detectar cambios en áreas pequeñas y realizar un seguimiento más detallado.

- Para abordar las limitaciones del modelo en la segmentación de lagunas con alta turbidez o con vegetación acuática superficial, se recomienda ampliar el conjunto de datos de entrenamiento con más muestras de cuerpos de agua que presenten estas características. Específicamente, se sugiere incluir imágenes de lagunas en diferentes zonas de la cordillera del Perú, donde estas condiciones son comunes.

- Se sugiere extender el modelo actual para abarcar no solo lagunas, sino también otros tipos de cuerpos de agua como ríos y embalses. Para ello, es importante enriquecer el conjunto de datos de entrenamiento con imágenes que incluyan estos elementos, permitiendo al modelo generalizar mejor y mejorar su aplicabilidad en diversas zonas geográficas.

- Conforme se disponga de más datos de campo o de fuentes confiables, se recomienda realizar una evaluación exhaustiva de la precisión del modelo mediante herramientas

estadísticas avanzadas. Esto contribuirá a validar y optimizar el modelo, asegurando que mantenga su precisión en diferentes contextos ambientales y con nuevos datos.

Bibliografía

Adrian, R., O'Reilly, C. M., Zagarese, H., Baines, S. B., Hessen, D. O., Keller, W., Livingstone, D. M., Sommaruga, R., Straile, D., Van Donk, E., et al. (2009). Lakes as sentinels of climate change. *Limnology and oceanography*, *54*(6part2), 2283-2297.

Amyar, A., Guo, R., Cai, X., Assana, S., Chow, K., Rodriguez, J., Yankama, T., Cirillo, J., Pierce, P., Goddu, B., et al. (2022). Impact of deep learning architectures on accelerated cardiac T1 mapping using MyoMapNet. *NMR in Biomedicine*, *35*(11), e4794.

Bishop, C. M. (2006). Pattern recognition and machine learning. *Springer google schola*, *2*, 1122-1128.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6), 679-698.

Cao, F., Yang, Z., Ren, J., Jiang, M., y Ling, W.-K. (2017). Does normalization methods play a role for hyperspectral image classification? *arXiv preprint arXiv:1710.02939*.

Chaurasia, A., y Culurciello, E. (2017). Linknet: Exploiting encoder representations for efficient semantic segmentation. *2017 IEEE visual communications and image processing (VCIP)*, 1-4.

Chen, C., Wang, Y., Yang, S., Ji, X., y Wang, G. (2023). A K-Net-based hybrid semantic segmentation method for extracting lake water bodies. *Engineering Applications of Artificial Intelligence*, *126*, 106904.

Corrales, M. (2023). Consultado el 24 de abril de 2024, desde https://larepublica.pe/sociedad/2023/11/21/cusco-por-que-la-laguna-de-piuray-que-abastece-de-agua-al-40-de-la-poblacion-de-cusco-debe-cerrar-lrsd-1688820

Dodds, W. K. (2002). *Freshwater ecology: concepts and environmental applications*. Elsevier.

Dozier, J., y Painter, T. H. (2004). Multispectral and hyperspectral remote sensing of alpine snow properties. *Annu. Rev. Earth Planet. Sci.*, *32*, 465-494.

Duda, R. O., y Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, *15*(1), 11-15.

Erdem, F., Bayram, B., Bakirman, T., Bayrak, O. C., y Akpinar, B. (2021). An ensemble deep learning based shoreline segmentation approach (WaterNet) from Landsat 8 OLI images. *Advances in Space Research*, *67*(3), 964-974.

Feng, Y., Jia, L., Zhang, J., y Chen, J. (2024). FFSwinNet: CNN-Transformer combined Network with FFT for Shale Core SEM Image Segmentation. *IEEE Access*.

Florez, R., Palomino-Quispe, F., Coaquira-Castillo, R. J., Herrera-Levano, J. C., Paixão, T., y Alvarez, A. B. (2023). A cnn-based approach for driver drowsiness detection by real-time eye state identification. *Applied Sciences*, *13*(13), 7849.

Garcia, M., Alcayaga, H., y Pizarro, A. (2023). Automatic segmentation of water bodies using rgb data: A physically based approach. *Remote Sensing*, *15*(5), 1170.

Ghosh, A., Ehrlich, M., Shah, S., Davis, L. S., y Chellappa, R. (2018). Stacked U-Nets for ground material segmentation in remote sensing imagery. *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 257-261.

Gonzales, R., y Woods, R. (2008). *Digital Image Processing*. USA: Prentice Hall.

Goodfellow, I., Bengio, Y., y Courville, A. (2016). *Deep Learning* [http://www.deeplearningbook.org]. MIT Press.

Haykin, S. S., et al. (2009). Neural networks and learning machines/Simon Haykin.

Hernández S., R. (2014). *Metodología de la Investigación*. McGrawHillEducation.

Huang, H., Lin, L., Tong, R., Hu, H., Zhang, Q., Iwamoto, Y., Han, X., Chen, Y.-W., y Wu, J. (2020). Unet 3+: A full-scale connected unet for medical image segmentation. *ICASSP 2020-2020 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 1055-1059.

INAIGEM. (2018). Consultado el 4 de agosto de 2024, desde https://www.gob.pe/institucion/inaigem/informes-publicaciones/783238-mapa-ubicacion-geografica-del-ambito-de-influencia-de-la-cordillera-vilcanota

INAIGEM. (2020a). Informe técnico de inspección: Laguna Yawarcocha, Cordiller del Vilcanota.

INAIGEM. (2020b). Inventario lagunas de origen glaciar 2020.

INAIGEM. (2020c). Reporte de Peligros en Glaciares RPG 001-2020, Lagunas en formación, cordillera del Vilcanota.

INAIGEM. (2023). Memoria Descriptiva del Inventario Nacional de Glaciares y Lagunas de Origen Glaciar 2023.

Jiang, C., Zhang, H., Wang, C., Ge, J., y Wu, F. (2022). Water Surface Mapping from Sentinel-1 Imagery Based on Attention-UNet3+: A Case Study of Poyang Lake Region. *Remote Sensing*, *14*(19), 4708.

Johnes, P., Moss, B., y Phillips, G. (1994). Lakes-Classification & Monitoring: A strategy for the classification of lakes.

Kaushik, S., Singh, T., Joshi, P. K., y Dietz, A. J. (2022). Automated mapping of glacial lakes using multisource remote sensing data and deep convolutional neural network. *International Journal of Applied Earth Observation and Geoinformation*, *115*, 103085.

King, A. P., y Eckersley, R. J. (2019). Chapter 6 - Inferential Statistics III: Nonparametric Hypothesis Testing. En A. P. King y R. J. Eckersley (Eds.), *Statistics for Biomedical Engineers and Scientists* (pp. 119-145). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-08-102939-8.00015-3

Kingma, D. P. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kriegler, F. J. (1969). Preprocessing transformations and their effects on multspectral recognition. *Proceedings of the Sixth International Symposium on Remote Sesning of Environment*, 97-131.

Krizhevsky, A., Sutskever, I., e Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

Labbihi, I., El Meslouhi, O., Benaddy, M., Kardouchi, M., y Akhloufi, M. (2024). Combining frequency transformer and CNNs for medical image segmentation. *Multimedia Tools and Applications*, *83*(7), 21197-21212.

Lachinov, D., Vasiliev, E., y Turlapov, V. (2018). Glioma segmentation with cascaded UNet. *International MICCAI Brainlesion Workshop*, 189-198.

Laura, K. R., y Alvarez, C. C. (2013). Identificación y Registro Catastral de Cuerpos de Agua mediante Técnicas de Procesamiento Digital de Imágenes. *arXiv preprint arXiv:1309.7609*.

Li, L., Yan, Z., Shen, Q., Cheng, G., Gao, L., y Zhang, B. (2019). Water body extraction from very high spatial resolution remote sensing data based on fully convolutional networks. *Remote Sensing*, *11*(10), 1162.

Li, Y., Dang, B., Zhang, Y., y Du, Z. (2022). Water body classification from high-resolution optical remote sensing imagery: Achievements and perspectives. *ISPRS Journal of Photogrammetry and Remote Sensing*, *187*, 306-327.

Lillesand, T., Kiefer, R. W., y Chipman, J. (2015). *Remote sensing and image interpretation*. John Wiley & Sons.

Liu, S., Wu, Y., Zhang, G., Lin, N., y Liu, Z. (2023). Comparing water indices for landsat data for automated surface water body extraction under complex ground background: a case study in Jilin Province. *Remote Sensing*, *15*(6), 1678.

Long, J., Shelhamer, E., y Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431-3440.

Luo, X., Tong, X., y Hu, Z. (2021). An applicable and automatic method for earth surface water mapping based on multispectral images. *International Journal of Applied Earth Observation and Geoinformation*, *103*, 102472.

Ma, C., Chen, Y., Hu, K., Du, C., Dong, J., y Lyu, L. (2024). Climate Warming Triggered a Glacial Lake Outburst Flood and Debris Flow Events in an Alpine Watershed, Western Himalayas, Tibet Plateau. *Bulletin of Engineering Geology and the Environment*, *83*(5), 201.

McFeeters, S. K. (1996). The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features. *International journal of remote sensing*, *17*(7), 1425-1432.

Motschmann, A. (2021). *Water resource risks in the Andes of Peru: an integrative perspective* [Tesis doctoral, University of Zurich].

Oppenheim, A. V., Willsky, A. S., Nawab, S. H., et al. (1998). *Señales y sistemas*. México: Prentice-Hall Hispanoamericana,

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, *9*(1), 62-66.

Patil, D. D., y Deore, S. G. (2013). Medical image segmentation: a review. *International Journal of Computer Science and Mobile Computing*, *2*(1), 22-27.

Perez-Torres, W. I., Uman-Flores, D. A., Quispe-Quispe, A. B., Palomino-Quispe, F., Bezerra, E., Leher, Q., Paixão, T., y Alvarez, A. B. (2024). Exploratory Analysis Using Deep Learning for Water-Body Segmentation of Peru's High-Mountain Remote Sensing Images. *Sensors*, *24*(16), 5177.

Rainio, O., Teuho, J., y Klén, R. (2024). Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, *14*(1), 6086.

Ronneberger, O., Fischer, P., y Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, 234-241.

Seekell, D., Cael, B., Lindmark, E., y Byström, P. (2021). The fractal scaling relationship for river inlets to lakes. *Geophysical Research Letters*, *48*(9), e2021GL093366.

Sharma, A., Thakur, V., Prakash, C., Sharma, A., y Sharma, R. (2024). Deep Learning-Based Glacial Lakes Extraction and Mapping in the Chandra–Bhaga Basin. *Journal of the Indian Society of Remote Sensing*, *52*(2), 435-447.

Terven, J., Cordova-Esparza, D. M., Ramirez-Pedraza, A., y Chavez-Urbiola, E. A. (2023). Loss functions and metrics in deep learning. A review. *arXiv preprint arXiv:2307.02694*.

UNP, U. N. d. l. P. (2024). Consultado el 24 de octubre de 2024, desde https://www.mate.unlp.edu.ar/practicas/117_20_19062013201933.pdf

USGS. (2022). *Landsat 8-9, Operational Land Imager (OLI) - Thermal Infrared Sensor (TIRS) Collection 2 (C2) Level 2 (L2) Data Format Control Book (DFCB)*.

USGS. (2024a). Consultado el 4 de agosto de 2024, desde https://www.usgs.gov/landsat-missions/landsat-7

USGS. (2024b). Consultado el 4 de agosto de 2024, desde https://www.usgs.gov/landsat-missions/landsat-level-1-processing-details

USGS. (2024c). Consultado el 4 de agosto de 2024, desde https://www.usgs.gov/landsat-missions/landsat-collection-2-level-2-science-products

USGS. (2024d). Consultado el 4 de agosto de 2024, desde https://www.usgs.gov/faqs/how-do-i-use-a-scale-factor-landsat-level-2-science-products

USGS. (2024e). Consultado el 4 de agosto de 2024, desde https://landsat.gsfc.nasa.gov/about/the-worldwide-reference-system/

Verschoof-Van der Vaart, W. B., y Lambers, K. (2019). Learning to look at LiDAR: The use of R-CNN in the automated detection of archaeological objects in LiDAR data from the Netherlands. *Journal of Computer Applications in Archaeology*, *2*(1).

Wang, H., Chen, X., Zhang, T., Xu, Z., y Li, J. (2022). CCTNet: Coupled CNN and transformer network for crop segmentation of remote sensing images. *Remote Sensing*, *14*(9), 1956.

Wang, S., Peppa, M. V., Xiao, W., Maharjan, S. B., Joshi, S. P., y Mills, J. P. (2022). A second-order attention network for glacial lake segmentation from remotely sensed imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, *189*, 289-301.

Wang, Y., Yang, K., Jia, T., y Luo, Y. (2023). Influence of natural factors and land use change on changes in the main lake area in China over the past 30 years. *Ecological Indicators*, *155*, 111005.

Wood, J., Harrison, S., Wilson, R., Emmer, A., Yarleque, C., Glasser, N., Torres, J., Caballero, A., Araujo, J., Bennett, G., et al. (2021). Contemporary glacial lakes in the Peruvian Andes. *Global and Planetary Change*, *204*, 103574.

Wu, J., Ji, W., Fu, H., Xu, M., Jin, Y., y Xu, Y. (2024). Medsegdiff-v2: Diffusion-based medical image segmentation with transformer. *Proceedings of the AAAI Conference on Artificial Intelligence*, *38*(6), 6030-6038.

Xu, H. (2006). Modification of normalised difference water index (NDWI) to enhance open water features in remotely sensed imagery. *International journal of remote sensing*, *27*(14), 3025-3033.

Yuan, K., Zhuang, X., Schaefer, G., Feng, J., Guan, L., y Fang, H. (2021). Deep-learning-based multispectral satellite image segmentation for water body detection. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *14*, 7422-7434.

Zhang, X., Li, J., y Hua, Z. (2022). MRSE-Net: multiscale residuals and SE-attention network for water body segmentation from satellite images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *15*, 5049-5064.

Zhang, Z., Liu, Q., y Wang, Y. (2018). Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, *15*(5), 749-753.

Zhao, H., Wang, S., Liu, X., y Chen, F. (2023). Exploring contrastive representation for weakly-supervised glacial lake extraction. *Remote Sensing*, *15*(5), 1456.

Zhao, H., Shi, J., Qi, X., Wang, X., y Jia, J. (2017). Pyramid scene parsing network. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2881-2890.

Zhou, S., Canchila, C., y Song, W. (2023). Deep learning-based crack segmentation for civil infrastructure: Data types, architectures, and benchmarked performance. *Automation in Construction*, *146*, 104678.

Zhou, Y., Yang, K., Ma, F., Hu, W., y Zhang, F. (2022). Water–land segmentation via structure-aware CNN–transformer network on large-scale SAR data. *IEEE Sensors Journal*, *23*(2), 1408-1422.

Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., y Liang, J. (2018). Unet++: A nested u-net architecture for medical image segmentation. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*, 3-11.

Apéndice A

Código de Procesamiento de Escenas Satelitales

```
# Genera los parches en formato TIF y ya en valores de reflectancia
import sys
import os
from glob import glob
import xml.etree.ElementTree as ET
import cv2
from scipy import ndimage
import matplotlib.pyplot as plt
import rasterio
import numpy as np
import math
from rasterio.plot import show
import matplotlib.patches as patches
# Verifica si se proporciono un argumento de linea de comandos
if len(sys.argv) != 2:
    print("Uso: python3 script.py <directorio_escena> ejem: LC08..._T1")
    sys.exit(1)
# Obtener el argumento de linea de comandos que representa la direccion y
    el path y row
############################
direccion = sys.argv[1]
path_row = direccion[10:16]
############################
def adjust_size(img, output_size):
  shape = img.shape
  x, y = shape[1], shape[0]
  if y < x:
    if y % output_size != 0:
      y = y- (y % output_size)
    return img[:y,:x-(x-y)]
  else:
    if x % output_size != 0:
      x = shape[1] - (x % output_size)
    return img[:y-(y-x),:x]
```

```
# Funcion para cortar escena
def crop_image(img=None, output_size=256):
  img_gray = img[:, :, 0]
  img_gray = cv2.GaussianBlur(img[:,:,0], (11, 11), 0)
  val, bin_mask = cv2.threshold(img_gray,1,255,cv2.THRESH_BINARY)
  edged = cv2.Canny(np.uint8(bin_mask), 10, 250, apertureSize=3)
  (cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL, cv2.
      CHAIN_APPROX_SIMPLE)
  idx = 0
  for c in cnts:
      x,y,w,h = cv2.boundingRect(c)
      if w>200 and h>200:
          idx+=1
          pad_x, pad_y = 100, 100
          new_img=img[y+pad_y:y+h-pad_y,x+pad_x:x+w-pad_x]
  new_img = adjust_size(new_img, output_size=output_size)
  return new_img
# Funcion para enderezar escena
def deskew_image(image_array):
    img_before = image_array[:, :, 0]
    img_before_gray = img_before.copy()
    val, bin_mask = cv2.threshold(np.uint8(img_before_gray),0,255,cv2.
        THRESH_BINARY)
    img_edges = cv2.Canny(np.uint8(bin_mask), 100, 100, apertureSize=3)
    lines = cv2.HoughLinesP(img_edges, 1, math.pi / 180.0, 100,
        minLineLength=100, maxLineGap=5)
    angles = []
    if (lines is not None):
        for x1, y1, x2, y2 in lines[0]:
          angle = math.degrees(math.atan2(y2 - y1, x2 - x1))
          angles.append(angle)
        median_angle = np.median(angles)
    if median_angle <= 0.0:
       median_angle = 90 + median_angle
    if (median_angle != 0):
          img_rotated = ndimage.rotate(image_array, median_angle, order=2)
    else:
          img_rotated = image_array
    img_crop = crop_image(img_rotated, output_size=256)
    # Retorna la imagen de n bandas recortada en un factor de 256
    return img_crop
def open_bands_and_deskew(list_bands, directory):
    # Cargando nombres de archivos de bandas de interes
    bands_files = []
    for band in list_bands:
        try:
            path = glob(f'../TRAIN_SCENES/{path_row}/{directory}/*_B{band}.
                TIF')[0]
            bands_files.append(path)
            print(f"Band␣{band}␣exists.")
        except IndexError:
            raise FileNotFoundError(f"No␣file␣found␣for␣band␣{band}.")
    # Intentando cargar archivo de metadatos
    metadata_filename = f"../TRAIN_SCENES/{path_row}/{directory}/*.xml"
    try:
        metadata_path = glob(metadata_filename)[0]
        print(f"Metadafile␣exists.")
    except IndexError:
```

```
88          raise FileNotFoundError(f"No metadata file found for this scene.")
89      # Cargando los archivos TIF
90      bands_digital_number = []
91      for band_name in bands_files:
92          band_array = rasterio.open(band_name)
93          bands_digital_number.append(band_array.read(1))
94          band_array.close()
95      print(f"{len(bands_digital_number)} bands have been read.")
96      # Apilando las bandas
97      stack_image = np.stack(bands_digital_number, axis=-1)
98      print(f"Stacked image has shape: {stack_image.shape}")
99      # Alineando imagen
100     print("Aligning image")
101     deskew = deskew_image(stack_image)
102     # Shape de la imagen de salida
103     print(f'Final shape of image is {deskew.shape}.')
104     # Obteniendo los valores de reflectancia
105     with open(metadata_path) as f:
106         # Cargando archivo metada.xml
107         tree = ET.parse(f)
108         root = tree.getroot()
109         MULT = float (root.find(f'.//REFLECTANCE_MULT_BAND_1').text)
110         ADD = float (root.find(f'.//REFLECTANCE_ADD_BAND_1').text)
111     return deskew * MULT + ADD
112
113 image = open_bands_and_deskew([2,3,4,5,6,7], direccion)
114 directorio_parches = f'../TRAIN_PATCHES/{path_row}/{direccion}'#{direccion}
        _parches/'
115 os.makedirs(directorio_parches, exist_ok=True)
116 alto = image.shape[0]
117 ancho = image.shape[1]
118 parche_size = 256
119 profile = {
120     'count' : image.shape[2],
121     'height' : parche_size,
122     'width' : parche_size,
123     'dtype' : 'float64'
124 }
125
126 if not os.path.exists(directorio_parches):
127     os.makedirs(directorio_parches)
128 for i, y in enumerate(range(0, alto, parche_size)):
129     for j, x in enumerate(range(0, ancho, parche_size)):
130         parche = image[y:y+parche_size, x:x+parche_size]
131         parche = parche.transpose(2, 0, 1)
132         nombre_parche = f"{direccion}_{i+1:03d}_{j+1:03d}.TIF"
133         ruta_parche = os.path.join(directorio_parches, nombre_parche)
134         with rasterio.open(ruta_parche, 'w', **profile) as dst:
135             dst.write(parche)
```

Apéndice B

Implementación de UNet con Tensorflow

```
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, BatchNormalization, Activation,
    MaxPooling2D, Dropout, Conv2DTranspose, concatenate, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Model
# Se define el modulo de Doble Convolucion
def conv2d_block(input_tensor, n_filters, kernel_size=3, batchnorm=True):
    # Define a convolutional block
    x = Conv2D(filters=n_filters, kernel_size=(kernel_size, kernel_size),
        kernel_initializer='he_normal', padding='same', use_bias=False)(
        input_tensor)
    if batchnorm:
        x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(filters=n_filters, kernel_size=(kernel_size, kernel_size),
        kernel_initializer='he_normal', padding='same', use_bias=False)(x)
    if batchnorm:
        x = BatchNormalization()(x)
    x = Activation('relu')(x)
    return x
# Se implementa el modelo UNet
def UNet(input_shape=(256, 256, 6), n_filters=64, final_activation='sigmoid
    ', dropout=0.1, batchnorm=True):
    # Define the UNET model with modifications for reconstruction
    input_img = Input(shape=input_shape)
    # Contracting Path
    c1 = conv2d_block(input_img, n_filters * 1, kernel_size=3, batchnorm=
        batchnorm)
    p1 = MaxPooling2D((2, 2))(c1)
    p1 = Dropout(dropout)(p1)
    c2 = conv2d_block(p1, n_filters * 2, kernel_size=3, batchnorm=batchnorm
        )
    p2 = MaxPooling2D((2, 2))(c2)
    p2 = Dropout(dropout)(p2)
    c3 = conv2d_block(p2, n_filters * 4, kernel_size=3, batchnorm=batchnorm
        )
    p3 = MaxPooling2D((2, 2))(c3)
    p3 = Dropout(dropout)(p3)
    c4 = conv2d_block(p3, n_filters * 8, kernel_size=3, batchnorm=batchnorm
        )
    p4 = MaxPooling2D((2, 2))(c4)
```

```
    p4 = Dropout(dropout)(p4)
    c5 = conv2d_block(p4, n_filters=n_filters * 16, kernel_size=3,
        batchnorm=batchnorm)
    # Expansive Path
    u6 = Conv2DTranspose(n_filters * 8, (3, 3), strides=(2, 2), padding='
        same')(c5)
    u6 = concatenate([u6, c4])
    u6 = Dropout(dropout)(u6)
    c6 = conv2d_block(u6, n_filters * 8, kernel_size=3, batchnorm=batchnorm
        )
    u7 = Conv2DTranspose(n_filters * 4, (3, 3), strides=(2, 2), padding='
        same')(c6)
    u7 = concatenate([u7, c3])
    u7 = Dropout(dropout)(u7)
    c7 = conv2d_block(u7, n_filters * 4, kernel_size=3, batchnorm=batchnorm
        )
    u8 = Conv2DTranspose(n_filters * 2, (3, 3), strides=(2, 2), padding='
        same')(c7)
    u8 = concatenate([u8, c2])
    u8 = Dropout(dropout)(u8)
    c8 = conv2d_block(u8, n_filters * 2, kernel_size=3, batchnorm=batchnorm
        )
    u9 = Conv2DTranspose(n_filters * 1, (3, 3), strides=(2, 2), padding='
        same')(c8)
    u9 = concatenate([u9, c1])
    u9 = Dropout(dropout)(u9)
    c9 = conv2d_block(u9, n_filters * 1, kernel_size=3, batchnorm=batchnorm
        )
    outputs = Conv2D(1, (1, 1), padding="same", activation=final_activation
        )(c9)  # Output with 6 channels for reconstruction
    model = Model(inputs=[input_img], outputs=[outputs])
    return model
if __name__ == '__main__':
    model = UNet(input_shape=(256, 256, 6), n_filters=64, final_activation=
        'sigmoid')
    model.summary()
    tf.keras.utils.plot_model(model, 'modelo_unet_ns.png', show_shapes=True
        )
```

Apéndice C

Implementación del modelo propuesto UNetFFT con Tensorflow

```
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, BatchNormalization, Activation,
    MaxPooling2D, Dropout, Conv2DTranspose, concatenate, Input
from tensorflow.keras.models import Model
# Definicion de Doble Convolucion
def conv2d_block(input_tensor, n_filters, kernel_size=3, batchnorm=True):
    x = Conv2D(filters=n_filters, kernel_size=(kernel_size, kernel_size),
        padding='same', use_bias=False)(input_tensor)
    if batchnorm:
        x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(filters=n_filters, kernel_size=(kernel_size, kernel_size),
        padding='same', use_bias=False)(x)
    if batchnorm:
        x = BatchNormalization()(x)
    x = Activation('relu')(x)
    return x
# Definicion de operacion multiplicacion punto a punto
def pixelwise_multiplication(tensor, weights):
    output = tensor * weights
    return output
# Definicion del modulo propuesto basado en filtros aprendibles en el
    dominio de la frecuencia
def fourier_combination(encoder_tensor, encoder_weights, decoder_tensor,
    decoder_weights):
    # Creando copias de los datos de entrada: shape (batch_size, H, W, C)
    encoder_input = tf.identity(encoder_tensor)
    decoder_input = tf.identity(decoder_tensor)
    # Reordenando los tensores: shape (batch_size, C, H, W)
    encoder_tensor = tf.transpose(encoder_tensor, perm=[0, 3, 1, 2])
    decoder_tensor = tf.transpose(decoder_tensor, perm=[0, 3, 1, 2])
    # FFT de las caracteristicas del encoder y decoder en sus dimensiones
        mas internas H, W
    encoder_fft = tf.signal.fft2d(tf.cast(encoder_tensor, tf.complex64))
    decoder_fft = tf.signal.fft2d(tf.cast(decoder_tensor, tf.complex64))
    # Tomando solo parte real de FFT
    encoder_fft_real = tf.math.real(encoder_fft)
    decoder_fft_real = tf.math.real(decoder_fft)
```

```
    # Se supone los pesos son reales y con shape (C, H, W), se adiciona una dimension mas para el batch_size (batch_size, C, H, W)
    encoder_weights = tf.expand_dims(encoder_weights, axis=0)
    decoder_weights = tf.expand_dims(decoder_weights, axis=0)
    # Se aplica la multiplicacion pixel a pixel, shape (batch_size, C, H, W)
    encoder_filtered = pixelwise_multiplication(encoder_fft_real, encoder_weights)
    decoder_filtered = pixelwise_multiplication(decoder_fft_real, decoder_weights)
    # Se concatenan las entradas fft con filtro y sin filtro, shape (batch_size, C*4, H, W)
    combined_fft_real = tf.concat([encoder_filtered, encoder_fft_real, decoder_filtered, decoder_fft_real], axis = 1)
    # Reordenando los datos a (batch_size, H, W, C*4) para aplicar correctamente la convolucion
    combined_fft_real = tf.transpose(combined_fft_real, perm=[0, 2, 3, 1])
    # Se aplica una convoluciin para obtener una combinaciin lineal pixel a pixel, se aumenta la cantidad de filtros x2 y luego se obtiene la cantidad de entrada
    combined_fft_real = Conv2D(filters=combined_fft_real.shape[-1]*2, kernel_size=(3, 3), padding='same', use_bias=False)(combined_fft_real)
    combined_fft_real = BatchNormalization()(combined_fft_real)
    combined_fft_real = Activation('relu')(combined_fft_real)
    combined_fft_real = Conv2D(filters=combined_fft_real.shape[-1]/8, kernel_size=(1, 1), padding='same', use_bias=False)(combined_fft_real)
    combined_fft_real = BatchNormalization()(combined_fft_real)
    combined_fft_real = Activation('relu')(combined_fft_real) # shape (batch_size, H, W, C)
    # Volviendo a combined_fft_real a tipo complex_64, shape (batch_size, H, W, C)
    combined_fft_complex = tf.complex(combined_fft_real, tf.zeros_like(combined_fft_real))
    # Reordenando para aplicar correctamente la IFFT, shape (batch_size, C, H, W)
    combined_fft_complex = tf.transpose(combined_fft_complex, perm=[0, 3, 1, 2])
    # Aplicando la IFFT y extrayendo el valor absoluto, shape (batch_size, C, H, W)
    combined_ifft = tf.signal.ifft2d(combined_fft_complex)
    combined_ifft = tf.abs(combined_ifft)
    # Volver a la disposiciin original de las dimensiones, shape (batch_size, H, W, C)
    combined_ifft = tf.transpose(combined_ifft, perm=[0, 2, 3, 1])
    return combined_ifft
# Implementacion del modelo UNetFFT con 32 filtros en el nivel 1
def UNet_FFT(input_shape=(256, 256, 6), n_filters=32, final_activation='sigmoid', dropout=0.1, batchnorm=True):
    input_img = Input(shape=input_shape)
    # Contracting Path
    c1 = conv2d_block(input_img, n_filters * 1, kernel_size=3, batchnorm=batchnorm)
    p1 = MaxPooling2D((2, 2))(c1)
    p1 = Dropout(dropout)(p1)
    c2 = conv2d_block(p1, n_filters * 2, kernel_size=3, batchnorm=batchnorm)
```

```
    p2 = MaxPooling2D((2, 2))(c2)
    p2 = Dropout(dropout)(p2)
    c3 = conv2d_block(p2, n_filters * 4, kernel_size=3, batchnorm=batchnorm
        )
    p3 = MaxPooling2D((2, 2))(c3)
    p3 = Dropout(dropout)(p3)
    c4 = conv2d_block(p3, n_filters * 8, kernel_size=3, batchnorm=batchnorm
        )
    p4 = MaxPooling2D((2, 2))(c4)
    p4 = Dropout(dropout)(p4)
    c5 = conv2d_block(p4, n_filters=n_filters * 16, kernel_size=3,
        batchnorm=batchnorm) # Cuello de botella
    # Aqui se inicializan los pesos de los filtros aprendibles para cada
        uno de los niveles del codificador y decodificador
    initializer = tf.keras.initializers.HeUniform()
    filter_01_enc = tf.Variable(initializer(shape=(n_filters * 8, c4.shape
        [1], c4.shape[2])), trainable=True)
    filter_01_dec = tf.Variable(initializer(shape=(n_filters * 8, c4.shape
        [1], c4.shape[2])), trainable=True)
    filter_02_enc = tf.Variable(initializer(shape=(n_filters * 4, c3.shape
        [1], c3.shape[2])), trainable=True)
    filter_02_dec = tf.Variable(initializer(shape=(n_filters * 4, c3.shape
        [1], c3.shape[2])), trainable=True)
    filter_03_enc = tf.Variable(initializer(shape=(n_filters * 2, c2.shape
        [1], c2.shape[2])), trainable=True)
    filter_03_dec = tf.Variable(initializer(shape=(n_filters * 2, c2.shape
        [1], c2.shape[2])), trainable=True)
    filter_04_enc = tf.Variable(initializer(shape=(n_filters * 1, c1.shape
        [1], c1.shape[2])), trainable=True)
    filter_04_dec = tf.Variable(initializer(shape=(n_filters * 1, c1.shape
        [1], c1.shape[2])), trainable=True)
    # Decodificador y Conexiones de salto
    u6 = Conv2DTranspose(n_filters * 8, (3, 3), strides=(2, 2), padding='
        same')(c5)
    u6 = fourier_combination(c4, filter_01_enc, u6, filter_01_dec)
    u6 = Dropout(dropout)(u6)
    c6 = conv2d_block(u6, n_filters * 8, kernel_size=3, batchnorm=batchnorm
        )
    u7 = Conv2DTranspose(n_filters * 4, (3, 3), strides=(2, 2), padding='
        same')(c6)
    u7 = fourier_combination(c3, filter_02_enc, u7, filter_02_dec)
    u7 = Dropout(dropout)(u7)
    c7 = conv2d_block(u7, n_filters * 4, kernel_size=3, batchnorm=batchnorm
        )
    u8 = Conv2DTranspose(n_filters * 2, (3, 3), strides=(2, 2), padding='
        same')(c7)
    u8 = fourier_combination(c2, filter_03_enc, u8, filter_03_dec)
    u8 = Dropout(dropout)(u8)
    c8 = conv2d_block(u8, n_filters * 2, kernel_size=3, batchnorm=batchnorm
        )
    u9 = Conv2DTranspose(n_filters * 1, (3, 3), strides=(2, 2), padding='
        same')(c8)
    u9 = fourier_combination(c1, filter_04_enc, u9, filter_04_dec)
    u9 = Dropout(dropout)(u9)
    c9 = conv2d_block(u9, n_filters * 1, kernel_size=3, batchnorm=batchnorm
        )
    outputs = Conv2D(1, (1, 1), padding="same", activation=final_activation
        )(c9)
```

```
    model = Model(inputs=[input_img], outputs=[outputs])
    return model

if __name__ == '__main__':
    model = UNet_FFT(input_shape=(256, 256, 6), n_filters=64,
        final_activation='sigmoid')
    model.summary()
    tf.keras.utils.plot_model(model, 'unet_fourier_python.png', show_shapes
        =True)
```

Apéndice D

Implementación de modelo de comparación FCN con Tensorflow

```
import tensorflow as tf
from tensorflow.keras import layers, Model
# Implementacion de modelo FCN con backbone VGG16
def FCN8(input_shape=(256, 256, 6), n_filters=64, final_activation="sigmoid
    ", kernel=3, pool_size=(2, 2), unpool_size=(2,2)):
    inputs = layers.Input(shape=input_shape)
    channels = input_shape[2]
    # Implementacion de Encoder VGG16
    conv_1 = layers.Conv2D(n_filters * 1, (kernel, kernel), padding="same",
        use_bias=False)(inputs)
    conv_1 = layers.BatchNormalization()(conv_1)
    conv_1 = layers.Activation("relu")(conv_1) # 256, 256, 64
    conv_2 = layers.Conv2D(n_filters * 1, (kernel, kernel), padding="same",
        use_bias=False)(conv_1)
    conv_2 = layers.BatchNormalization()(conv_2)
    conv_2 = layers.Activation("relu")(conv_2) # 256, 256, 64
    pool_1 = layers.MaxPool2D(pool_size)(conv_2) # 128, 128, 64
    conv_3 = layers.Conv2D(n_filters * 2, (kernel, kernel), padding="same",
        use_bias=False)(pool_1)
    conv_3 = layers.BatchNormalization()(conv_3)
    conv_3 = layers.Activation("relu")(conv_3) # 128, 128, 128
    conv_4 = layers.Conv2D(n_filters * 2, (kernel, kernel), padding="same",
        use_bias=False)(conv_3)
    conv_4 = layers.BatchNormalization()(conv_4)
    conv_4 = layers.Activation("relu")(conv_4) # 128, 128, 128
    pool_2 = layers.MaxPool2D(pool_size)(conv_4) # 64, 64, 128
    conv_5 = layers.Conv2D(n_filters * 4, (kernel, kernel), padding="same",
        use_bias=False)(pool_2)
    conv_5 = layers.BatchNormalization()(conv_5)
    conv_5 = layers.Activation("relu")(conv_5) # 64, 64, 256
    conv_6 = layers.Conv2D(n_filters * 4, (kernel, kernel), padding="same",
        use_bias=False)(conv_5)
    conv_6 = layers.BatchNormalization()(conv_6)
    conv_6 = layers.Activation("relu")(conv_6) # 64, 64, 256
    conv_7 = layers.Conv2D(n_filters * 4, (kernel, kernel), padding="same",
        use_bias=False)(conv_6)
    conv_7 = layers.BatchNormalization()(conv_7)
    conv_7 = layers.Activation("relu")(conv_7) # 64, 64, 256
    pool_3 = layers.MaxPool2D(pool_size)(conv_7) # 32, 32, 256
```

```
    ##################
    skip_1 = layers.Conv2D(n_filters, (kernel, kernel), padding="same",
        use_bias=False)(pool_3) # 32, 32, 6
    skip_1 = layers.BatchNormalization()(skip_1)
    skip_1 = layers.Activation("relu")(skip_1) # 32, 32, 6
    ##################
    conv_8 = layers.Conv2D(n_filters * 8, (kernel, kernel), padding="same",
         use_bias=False)(pool_3)
    conv_8 = layers.BatchNormalization()(conv_8)
    conv_8 = layers.Activation("relu")(conv_8) # 32, 32, 512
    conv_9 = layers.Conv2D(n_filters * 8, (kernel, kernel), padding="same",
         use_bias=False)(conv_8)
    conv_9 = layers.BatchNormalization()(conv_9)
    conv_9 = layers.Activation("relu")(conv_9) # 32, 32, 512
    conv_10 = layers.Conv2D(n_filters * 8, (kernel, kernel), padding="same"
        , use_bias=False)(conv_9)
    conv_10 = layers.BatchNormalization()(conv_10)
    conv_10 = layers.Activation("relu")(conv_10) # 32, 32, 512
    pool_4 = layers.MaxPool2D(pool_size)(conv_10) # 16, 16, 512
    ##################
    skip_2 = layers.Conv2D(n_filters, (kernel, kernel), padding="same",
        use_bias=False)(pool_4) # 16, 16, 6
    skip_2 = layers.BatchNormalization()(skip_2)
    skip_2 = layers.Activation("relu")(skip_2) # 16, 16, 6
    ##################
    conv_11 = layers.Conv2D(n_filters * 8, (kernel, kernel), padding="same"
        , use_bias=False)(pool_4)
    conv_11 = layers.BatchNormalization()(conv_11)
    conv_11 = layers.Activation("relu")(conv_11) # 16, 16, 512
    conv_12 = layers.Conv2D(n_filters * 8, (kernel, kernel), padding="same"
        , use_bias=False)(conv_11)
    conv_12 = layers.BatchNormalization()(conv_12)
    conv_12 = layers.Activation("relu")(conv_12) # 16, 16, 512
    conv_13 = layers.Conv2D(n_filters * 8, (kernel, kernel), padding="same"
        , use_bias=False)(conv_12)
    conv_13 = layers.BatchNormalization()(conv_13)
    conv_13 = layers.Activation("relu")(conv_13) # 16, 16, 512
    pool_5 = layers.MaxPool2D(pool_size)(conv_13) # 8, 8, 512
    conv_14 = layers.Conv2D(n_filters * 16, (kernel, kernel), padding="same
        ", use_bias=False)(pool_5) # 8, 8, 1024
    conv_14 = layers.BatchNormalization()(conv_14)
    conv_14 = layers.Activation("relu")(conv_14)
    # Implementacion Decoder
    conv_15 = layers.Conv2D(n_filters * 16, (kernel, kernel), padding="same
        ", use_bias=False)(conv_14) # 8, 8, 1024
    conv_15 = layers.BatchNormalization()(conv_15)
    conv_15 = layers.Activation("relu")(conv_15)
    conv_16 = layers.Conv2D(n_filters, (kernel, kernel), padding="same",
        use_bias=False)(conv_15) # 8, 8, 64
    conv_16 = layers.BatchNormalization()(conv_16)
    conv_16 = layers.Activation("relu")(conv_16)
    ups_1 = layers.Conv2DTranspose(n_filters, kernel_size=(kernel, kernel),
         strides=(2, 2), padding='same', use_bias=False)(conv_16) # 16, 16,
        64
    ups_1 = layers.BatchNormalization()(ups_1)
    ups_1 = layers.Activation("relu")(ups_1)
    suma1 = layers.Add()([ups_1, skip_2]) # 16, 16, 64
    suma1 = layers.BatchNormalization()(suma1)
```

```
77      suma1 = layers.Activation("relu")(suma1)
78      ups_2 = layers.Conv2DTranspose(n_filters, kernel_size=(kernel, kernel),
            strides=(2, 2), padding='same', use_bias=False)(suma1) # 32, 32, 64
79      ups_2 = layers.BatchNormalization()(ups_2)
80      ups_2 = layers.Activation("relu")(ups_2)
81      suma2 = layers.Add()([ups_2, skip_1]) # 32, 32, 64
82      suma2 = layers.BatchNormalization()(suma2)
83      suma2 = layers.Activation("relu")(suma2)
84      outputs = layers.Conv2DTranspose(n_filters, kernel_size=(8, 8), strides
            =(8, 8), padding='same', use_bias=False)(suma2) # 256, 256, 64
85      outputs = layers.BatchNormalization()(outputs)
86      outputs = layers.Activation("relu")(outputs)
87      outputs = layers.Conv2D(1, (1, 1), padding="same", activation=
            final_activation)(outputs)
88      model = Model(inputs=inputs, outputs=outputs, name="FCN8")
89      return model
90  if __name__ == '__main__':
91      model = FCN8(input_shape=(256, 256, 6), n_filters=64, final_activation=
            "sigmoid")
92      model.summary()
93      tf.keras.utils.plot_model(model, 'modelo_fcn8.png', show_shapes=True)
```

Apéndice E

Implementación de modelo de comparación Linknet con Tensorflow

```
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, BatchNormalization, Activation,
    MaxPooling2D, Dropout, Conv2DTranspose, concatenate, Input, Add,
   UpSampling2D
from tensorflow.keras.models import Model
# Definiendo bloque codificador
def encoder_block(input_tensor, n_filters, kernel_size=3, strides=2):
    shortcut = input_tensor
    x = Conv2D(filters=n_filters, kernel_size=(kernel_size, kernel_size),
       strides=strides,kernel_initializer='he_normal', padding='same',
       use_bias=False)(input_tensor)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(filters=n_filters, kernel_size=(kernel_size, kernel_size),
       kernel_initializer='he_normal', padding='same', use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    shortcut = Conv2D(filters=n_filters, kernel_size=(1, 1), strides=
       strides, use_bias=False)(shortcut)
    shortcut = BatchNormalization()(shortcut)
    suma_1 = Add()([shortcut, x])
    x = Conv2D(filters=n_filters, kernel_size=(kernel_size, kernel_size),
       kernel_initializer='he_normal', padding='same', use_bias=False)(
       suma_1)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(filters=n_filters, kernel_size=(kernel_size, kernel_size),
       kernel_initializer='he_normal', padding='same', use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    suma_2 = Add()([suma_1, x])
    return suma_2
# Definiendo bloque decodificador
def decoder_block(input_tensor, n_filters, kernel_size=3, strides=2):
    filters_input = input_tensor.shape[2]
    x = Conv2D(filters=filters_input//4, kernel_size=(1, 1),
       kernel_initializer='he_normal', padding='same', use_bias=False)(
       input_tensor)
    x = BatchNormalization()(x)
```

```
    x = Activation('relu')(x)
    #Full-Conv
    #x = Conv2D(filters=filters_input//4, kernel_size=(3, 3),
        kernel_initializer='he_normal', padding='same', use_bias=False)(x)
    x = Conv2DTranspose(filters=filters_input//4, kernel_size=(3, 3),
        strides=strides, kernel_initializer='he_normal', padding='same',
        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(filters=n_filters, kernel_size=(1, 1), kernel_initializer='
        he_normal', padding='same', use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    return x
# Implementando modelo Linknet
def LinkNet(input_shape=(256, 256, 6), n_filters=64, final_activation='
    sigmoid'):
    input_img = Input(shape=input_shape) #256, 256, 6
    # Capa conv inicial
    x = Conv2D(filters=n_filters, kernel_size=(7, 7), strides=(2, 2),
        padding='same')(input_img) #128, 128, 64
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same')(x) #
        64, 64, 64
    #Encoder_block_1
    skip_1 = encoder_block(input_tensor=x, n_filters=n_filters, strides=1)
        # 64, 64, 64
    #Encoder_block_2
    skip_2 = encoder_block(input_tensor=skip_1, n_filters=n_filters*2) #
        32, 32, 128
    #Encoder_block_3
    skip_3 = encoder_block(input_tensor=skip_2, n_filters=n_filters*4) #
        16, 16, 256
    #Encoder_block_4
    skip_4 = encoder_block(input_tensor=skip_3, n_filters=n_filters*8) # 8,
         8, 512
    # Decoder cuarto nivel
    dec_4 = decoder_block(input_tensor=skip_4, n_filters=n_filters*4) # 16,
         16, 256
    #Suma_3
    parcial_4 = Add()([dec_4, skip_3])
    # Decoder tercer nivel
    dec_3 = decoder_block(input_tensor=parcial_4, n_filters=n_filters*2) #
        32, 32, 128
    #Suma_2
    parcial_3 = Add()([dec_3, skip_2])
    # Decoder segundo nivel
    dec_2 = decoder_block(input_tensor=parcial_3, n_filters=n_filters) #
        64, 64, 64
    #Suma_2
    parcial_2 = Add()([dec_2, skip_1])
    # Decoder primer nivel
    dec_1 = decoder_block(input_tensor=parcial_2, n_filters=n_filters,
        strides=1) # 64, 64, 64
    # Capas finales full-conv y conv
    #Full-Conv
```

```
71      x = Conv2DTranspose(filters=n_filters, kernel_size=(3, 3), strides=2,
            kernel_initializer='he_normal', padding='same', use_bias=False)(
            dec_1)
72      x = BatchNormalization()(x)
73      x = Activation('relu')(x) # 128, 128, 32
74      # Conv
75      x = Conv2D(filters=n_filters, kernel_size=(3, 3), kernel_initializer='
            he_normal', padding='same', use_bias=False)(x)
76      x = BatchNormalization()(x)
77      x = Activation('relu')(x) # 128, 128, 32
78      #Full-Conv
79      x = Conv2DTranspose(filters=n_filters, kernel_size=(2, 2), strides=2,
            kernel_initializer='he_normal', padding='same', use_bias=False)(x)
80      x = BatchNormalization()(x)
81      x = Activation('relu')(x) # 256, 256, 32
82      outputs = Conv2D(1, (1, 1), padding='same', activation=final_activation
            )(x) # 256, 256, 6
83      model = Model(inputs=[input_img], outputs=[outputs])
84      return model
85  if __name__ == '__main__':
86      model = LinkNet(input_shape=(256, 256, 6), n_filters=32,
            final_activation='sigmoid')
87      model.summary()
88
89      tf.keras.utils.plot_model(model, 'modelo_linknet_ns.png', show_shapes=
            True)
```

Apéndice F

Implementación de modelo de comparación PSPNet con Tensorflow

```
from tensorflow.keras.layers import Input, Conv2D, BatchNormalization,
    Activation, MaxPooling2D, GlobalAveragePooling2D, Add, Reshape,
    AveragePooling2D, Concatenate, Dropout, Conv2DTranspose
from tensorflow.keras import Model
import tensorflow as tf
### Modelo ResNet50
def conv_block(x, filters, kernel_size, strides=(1, 1), padding='same',
    dilation_rate=(1, 1)):
    x = Conv2D(filters=filters, kernel_size=kernel_size, strides=strides,
        padding=padding, dilation_rate=dilation_rate, use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    return x
def identity_block(x, filters, dilation_rate=(1, 1)):
    shortcut = x
    x = conv_block(x, filters=filters, kernel_size=(1, 1), strides=(1, 1),
        dilation_rate=(1, 1))
    x = conv_block(x, filters=filters, kernel_size=(3, 3), strides=(1, 1),
        dilation_rate=dilation_rate)
    x = Conv2D(filters=filters * 4, kernel_size=(1, 1), strides=(1, 1),
        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Add()([x, shortcut])
    x = Activation('relu')(x)
    return x
def projection_block(x, filters, strides=(1, 1), dilation_rate=(1, 1)):
    shortcut = x
    x = conv_block(x, filters=filters, kernel_size=(1, 1), strides=strides,
         dilation_rate=(1, 1))
    x = conv_block(x, filters=filters, kernel_size=(3, 3), strides=(1, 1),
        dilation_rate=dilation_rate)
    x = Conv2D(filters=filters * 4, kernel_size=(1, 1), strides=(1, 1),
        use_bias=False)(x)
    x = BatchNormalization()(x)
    shortcut = Conv2D(filters=filters * 4, kernel_size=(1, 1), strides=
        strides, use_bias=False)(shortcut)
    shortcut = BatchNormalization()(shortcut)
    x = Add()([x, shortcut])
    x = Activation('relu')(x)
```

```
    return x
# Implementando Resnet50 como backbone
def ResNet50(inputs, n_filters=64):
    filter_1 = n_filters
    filter_2 = n_filters
    filter_3 = n_filters * 2
    filter_4 = n_filters * 4
    filter_5 = n_filters * 8
    # Initial conv layer
    x = conv_block(inputs, filter_1, kernel_size=(7, 7), strides=(2, 2),
        padding='same')
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same')(x)
    # conv block 1
    x = projection_block(x, filter_2, strides=(1, 1))
    x = identity_block(x, filter_2)
    x = identity_block(x, filter_2)
    # conv block 2
    x = projection_block(x, filter_3, strides=(2, 2))
    x = identity_block(x, filter_3)
    x = identity_block(x, filter_3)
    x = identity_block(x, filter_3)
    # conv block 3 (with dilation to maintain spatial dimensions)
    x = projection_block(x, filter_4, strides=(1, 1), dilation_rate=(2, 2))
    x = identity_block(x, filter_4, dilation_rate=(2, 2))
    x = identity_block(x, filter_4, dilation_rate=(2, 2))
    x = identity_block(x, filter_4, dilation_rate=(2, 2))
    x = identity_block(x, filter_4, dilation_rate=(2, 2))
    x = identity_block(x, filter_4, dilation_rate=(2, 2))
    # conv block 4 (with higher dilation to maintain spatial dimensions)
    x = projection_block(x, filter_5, strides=(1, 1), dilation_rate=(4, 4))
    x = identity_block(x, filter_5, dilation_rate=(4, 4))
    x = identity_block(x, filter_5, dilation_rate=(4, 4))
    return x
### Modulo Pyramid Pooling
def ppm_output(input_tensor, input_dim=2048, reduction_dim=512,
    final_activation='sigmoid'):
    height = tf.shape(input_tensor)[1]
    width = tf.shape(input_tensor)[2]
    # Global Pooling
    global_pooling = GlobalAveragePooling2D()(input_tensor)
    global_pooling = Reshape((1, 1, input_dim))(global_pooling) # 1, 1,
        2048
    global_pooling = Conv2D(reduction_dim, (1, 1), use_bias=False)(
        global_pooling)
    global_pooling = BatchNormalization()(global_pooling)
    global_pooling = Activation('relu')(global_pooling)
    global_pooling = tf.image.resize(global_pooling, size=(height, width),
        method=tf.image.ResizeMethod.BILINEAR) # 32, 32, 512
    # Pooling 2x2
    pooling_2x2 = AveragePooling2D(pool_size=(16, 16), strides=(16, 16))(
        input_tensor) # 2, 2, 2048
    pooling_2x2 = Conv2D(reduction_dim, (1, 1), use_bias=False)(pooling_2x2
        )
    pooling_2x2 = BatchNormalization()(pooling_2x2)
    pooling_2x2 = Activation('relu')(pooling_2x2)
    pooling_2x2 = tf.image.resize(pooling_2x2, size=(height, width), method
        =tf.image.ResizeMethod.BILINEAR) # 32, 32, 512
    # Pooling 4x4
```

```
    pooling_4x4 = AveragePooling2D(pool_size=(8, 8), strides=(8, 8))(
        input_tensor) # 4, 4, 2048
    pooling_4x4 = Conv2D(reduction_dim, (1, 1), use_bias=False)(pooling_4x4
        )
    pooling_4x4 = BatchNormalization()(pooling_4x4)
    pooling_4x4 = Activation('relu')(pooling_4x4)
    pooling_4x4 = tf.image.resize(pooling_4x4, size=(height, width), method
        =tf.image.ResizeMethod.BILINEAR) # 32, 32, 512
    # Pooling 8x8
    pooling_8x8 = AveragePooling2D(pool_size=(4, 4), strides=(4, 4))(
        input_tensor) # 8, 8, 2048
    pooling_8x8 = Conv2D(reduction_dim, (1, 1), use_bias=False)(pooling_8x8
        )
    pooling_8x8 = BatchNormalization()(pooling_8x8)
    pooling_8x8 = Activation('relu')(pooling_8x8)
    pooling_8x8 = tf.image.resize(pooling_8x8, size=(height, width), method
        =tf.image.ResizeMethod.BILINEAR) # 32, 32, 512
    # Concatenando todas las salidas
    poolings = Concatenate(axis=3)([input_tensor, global_pooling,
        pooling_2x2, pooling_4x4, pooling_8x8]) # 32, 32, 4096
    poolings = Conv2D(reduction_dim, (3, 3), padding='same', use_bias=False
        )(poolings) # 32, 32, 512
    poolings = BatchNormalization()(poolings)
    poolings = Activation('relu')(poolings)
    poolings = Dropout(0.1)(poolings)
    poolings = Conv2D(reduction_dim // 8, (1, 1), padding='same', use_bias=
        False)(poolings) # 32, 32, 64
    poolings = BatchNormalization()(poolings)
    poolings = Activation('relu')(poolings)
    outputs = Conv2DTranspose(filters=reduction_dim // 8, kernel_size=(8,
        8), strides=(8, 8),padding='same', use_bias=False)(poolings) # 256,
        256, 64
    outputs = BatchNormalization()(outputs)
    outputs = Activation("relu")(outputs)
    outputs = Conv2D(filters=1, kernel_size=(1, 1), padding="same",
        activation=final_activation)(outputs)
    return outputs
# Implementando modelo PSPNet
def PSPNet(input_shape=(256, 256, 6), n_filters=64, final_activation='
    sigmoid'):
    inputs = Input(shape=input_shape)
    output_resnet50 = ResNet50(inputs, n_filters)
    output_ppm = ppm_output(input_tensor=output_resnet50,
        input_dim=output_resnet50.shape[3],
        reduction_dim=output_resnet50.shape[3]//4,
        final_activation=final_activation
    )
    model = Model(inputs, output_ppm)
    return model
if __name__ == '__main__':
    model = PSPNet(input_shape=(256, 256, 6), n_filters=64,
        final_activation='sigmoid')
    model.summary()
    tf.keras.utils.plot_model(model, 'modelo_PSPNet.png', show_shapes=True)
```

Apéndice G

Código para Entrenar Todos los Modelos

```
import os
import sys
from time import time
import argparse
# Adicionando la carpeta raiz del proyecto al PYTHONPATH
project_root = os.path.abspath(os.path.join(os.path.dirname(__file__), '../..'))
sys.path.append(project_root)
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import tensorflow as tf
from b_tfrecord_reader import ReadDataset, img_augmentation, count_elements_in_tfrecord
import pickle
import numpy as np
import matplotlib.pyplot as plt
# Importando los modelos para entrenamiento tanto de comparacion como el propuesto
from MODELS.UNet import UNet
from MODELS.LinkNet import LinkNet
from MODELS.PSPNet import PSPNet
from MODELS.FCN8 import FCN8
from MODELS.UNet_plus_FFT import UNet_FFT
############
# Configuracion del analizador de argumentos
parser = argparse.ArgumentParser(description="Script de entrenamiento para diferentes modelos de segmentacion.")
parser.add_argument('model', choices=['unet_32', 'unet_64', 'linknet_32', 'linknet_64', 'pspnet_32', 'pspnet_64', 'fcn8_32', 'fcn8_64', 'unet_fft_16', 'unet_fft32'], help='Modelo a utilizar para el entrenamiento')
# Parsear los argumentos
args = parser.parse_args()
# Importar el archivo de configuracion correspondiente basado en el argumento
if args.model == 'unet_32':
    import config_unet_32 as config
elif args.model == 'unet_64':
    import config_unet_64 as config
elif args.model == 'linknet_32':
    import config_linknet_32 as config
elif args.model == 'linknet_64':
```

```
    import config_linknet_64 as config
elif args.model == 'pspnet_32':
    import config_pspnet_32 as config
elif args.model == 'pspnet_64':
    import config_pspnet_64 as config
elif args.model == 'fcn8_32':
    import config_fcn8_32 as config
elif args.model == 'fcn8_64':
    import config_fcn8_64 as config
elif args.model == 'unet_fft_16':
    import config_unet_fft_16 as config
elif args.model == 'unet_fft_32':
    import config_unet_fft_32 as config
else:
    raise ValueError("Modelo no soportado. Elija entre 'unet_32', 'unet_64', 'linknet_32', 'linknet_64', 'pspnet_32', 'pspnet_64', 'fcn8_32', 'fcn8_64', 'unet_fft_16', 'unet_fft_32'.")
##############
def train_ds_batches(tfrecord_path, batch_size=4, buffersize=1000):
    LENGTH = count_elements_in_tfrecord(tfrecord_path)
    STEPS_PER_EPOCH = LENGTH // batch_size
    train_dataset = ReadDataset(tfrecord_path)
    train_batches = (
        train_dataset
        .shuffle(buffersize)
        .batch(batch_size)
        .map(img_augmentation)
        .repeat()
        .prefetch(buffer_size=tf.data.AUTOTUNE)
        )
    return train_batches, STEPS_PER_EPOCH
def val_test_ds_batches(tfrecord_path, batch_size=4):
    LENGTH = count_elements_in_tfrecord(tfrecord_path)
    STEPS_PER_EPOCH = LENGTH // batch_size
    val_test_dataset = ReadDataset(tfrecord_path)
    val_test_batches = (
        val_test_dataset
        .cache()
        .batch(batch_size)
        .prefetch(buffer_size=tf.data.AUTOTUNE)
    )
    return val_test_batches, STEPS_PER_EPOCH
############
# Se define la metrica de evaluacion MIoU para dos clases
class miou_binary(tf.keras.metrics.MeanIoU):
    def update_state(self, y_true, y_pred, sample_weight=None):
        y_pred = tf.where(y_pred>0.5, 1, 0)
        super().update_state(y_true, y_pred, sample_weight)
############
if __name__ == '__main__':
    print(f"ENTRENAMIENTO DE TAREA SEGMENTACION - MODELO: {config.model}")
    print("============================")
    print(f"DATASET OBTENIDO DE: {config.dataset_dir}")
    print(f"MEJOR MODELO GUARDADO EN: {config.best_model_dir}")
    print(f"NOMBRE DEL ARCHIVO KERAS: {config.model_name}")
    print(f"NOMBRE DEL ARCHIVO PKL: {config.history_name}")
    print(f"ENTRENADO DURANTE {config.epocas} EPOCAS, PACIENCIA DE {config.paciencia}, BATCH SIZE DE {config.batch_size}")
```

```
    print("============================")
    ###########
    # Cargando los archivos de entrenamiento y validacion
    train_tfrecord_path = os.path.join(config.dataset_dir, 'train_ds.tfrecords')
    val_tfrecord_path = os.path.join(config.dataset_dir, 'val_ds.tfrecords')
    train_batches, steps_train = train_ds_batches(train_tfrecord_path, batch_size=config.batch_size, buffersize=1000)
    val_batches, steps_val = val_test_ds_batches(val_tfrecord_path, batch_size=config.batch_size)
    ###########
    # Crear la carpeta para guardar los modelos si no existe y carpeta para guardar las imagenes de entrenamiento
    if not os.path.exists(config.best_model_dir):
        os.makedirs(config.best_model_dir)
    if not os.path.exists(config.on_epoch_end_dir):
        os.makedirs(config.on_epoch_end_dir)
    ###########
    # Seleccionando modelo a entrenar
    if config.model == 'unet_32':
        model = UNet(input_shape=(256, 256, 6), n_filters=32, final_activation='sigmoid')
    elif config.model == 'unet_64':
        model = UNet(input_shape=(256, 256, 6), n_filters=64, final_activation='sigmoid')
    elif config.model == 'linknet_32':
        model = LinkNet(input_shape=(256, 256, 6), n_filters=32, final_activation='sigmoid')
    elif config.model == 'linknet_64':
        model = LinkNet(input_shape=(256, 256, 6), n_filters=64, final_activation='sigmoid')
    elif config.model == 'pspnet_32':
        model = PSPNet(input_shape=(256, 256, 6), n_filters=32, final_activation='sigmoid')
    elif config.model == 'pspnet_64':
        model = PSPNet(input_shape=(256, 256, 6), n_filters=64, final_activation='sigmoid')
    elif config.model == 'fcn8_32':
        model = FCN8(input_shape=(256, 256, 6), n_filters=32, final_activation='sigmoid')
    elif config.model == 'fcn8_64':
        model = FCN8(input_shape=(256, 256, 6), n_filters=64, final_activation='sigmoid')
    elif config.model == 'unet_fft_16':
        model = UNet_FFT(input_shape=(256, 256, 6), n_filters=16, final_activation='sigmoid')
    elif config.model == 'unet_fft_32':
        model = UNet_FFT(input_shape=(256, 256, 6), n_filters=32, final_activation='sigmoid')
    else:
        raise ValueError(f"config.model: {config.model} no esta en los modelos disponibles.")
    print(f"Parametros del modelo: {model.count_params()}")
    print(f"Peso del modelo en MB: {model.count_params()* 4 / (1024 * 1024)}")
    print("============================")
    model.summary()
```

```
130     tf.keras.utils.plot_model(model, f'./{config.best_model_dir}/{config.
            model_name}.png', show_shapes=True)
131     print("============================")
132     print("Compilando␣modelo␣con␣optimizador␣Adam,␣perdida␣de␣
            BinaryCrossEntropy␣y␣metricas␣de␣evaluacion␣miou_binary␣y␣
            binary_accuracy.")
133     # Compilando modelo a entrenar
134     model.compile(optimizer='adam', loss='binary_crossentropy', metrics=[
            miou_binary(num_classes=2), 'binary_accuracy'])
135     ############
136     # Creando Callbacks para el entrenamiento: predecir al final de epocas,
            guardar mejor modelo, early stopping
137     # Creando clase Callback para guardar una imagen de muestra cada 4
            epocas
138     class PredictionCallback(tf.keras.callbacks.Callback):
139         def __init__(self, validation_data, model):
140             super(PredictionCallback, self).__init__()
141             self.validation_data = validation_data
142             self.model = model
143         def on_epoch_end(self, epoch, logs=None):
144             # Si la epoca es multiplo de 2, genera la prediccion de una
                    imagen
145             if epoch % 2 == 0:
146                 for img, mask in self.validation_data.take(1):
147                     val_pred = self.model.predict(img)
148                 val_pred = np.array(val_pred[1])
149                 plt.imsave(f'{config.on_epoch_end_dir}/{config.model}_{
                        epoch:03d}.png', val_pred.squeeze(), cmap='viridis')
150             if epoch == 0:
151                 for _, mask in self.validation_data.take(1):
152                     mask_img = np.array(mask[1])
153                     plt.imsave(f'{config.on_epoch_end_dir}/{config.model}
                            _original.png', mask_img.squeeze(), cmap='viridis')
154     # Callback para guardar el mejor modelo
155     checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
156         filepath=config.model_name,
157         monitor='val_loss',
158         save_best_only=True,
159         save_weights_only=False,
160         mode='auto',
161         save_freq='epoch'
162     )
163     # Callback para detener el entrenamiento si no hay mejora
164     early_stopping_callback = tf.keras.callbacks.EarlyStopping(
165         monitor='val_loss',
166         patience=config.paciencia,  # Numero de epocas sin mejora para
                detener
167         verbose=1
168     )
169     prediction_callback = PredictionCallback(
170         validation_data=val_batches,
171         model=model,
172         )
173     ############
174     # Inicio de entrenamiento
175     print("================================")
176     start_time = time()
177     print(f"El␣entrenamiento␣comenzo:␣{start_time}")
```

```
    # Entrenando modelo
    history = model.fit(
        train_batches,
        epochs=config.epocas,
        steps_per_epoch=steps_train,
        validation_data=val_batches,
        validation_steps=steps_val,
        callbacks=[checkpoint_callback, early_stopping_callback,
            prediction_callback],
        verbose=2,
    )
    end_time = time()
    print(f"El entrenamiento finalizo: {end_time}")
    print(f"El entrenamiento duro: {(end_time - start_time)/60} minutos.")
    # Guarda el history en un archivo
    with open(config.history_name, 'wb') as f:
        pickle.dump(history.history, f)
    print("Final")
```

Apéndice H

Código para Realizar la Prueba de Wilcoxon para Muestras Pareadas

```python
from metrics import calculate_metrics_listed
from glob import glob
import rasterio
import numpy as np
from scipy.stats import  norm
from scipy.stats import wilcoxon

def metrics_model_listed(preds_dir, masks_dir):
    dir_nombres_pred = sorted(glob(f"{preds_dir}/*"))
    dir_nombres_masks = sorted(glob(f"{masks_dir}/*"))
    preds = []
    masks = []
    for pred_dir, mask_dir in zip(dir_nombres_pred, dir_nombres_masks):
        prediction = rasterio.open(pred_dir).read(1)
        prediction = np.where(prediction > 0.5, 1, 0)
        preds.append(prediction)
        mask = rasterio.open(mask_dir).read(1)
        mask = np.where(mask > 0.5, 1, 0)
        masks.append(mask)
    preds = np.array(preds)
    masks = np.array(masks)
    # Calculando mericas
    miou_list, pix_acc_list, F1_s_list = calculate_metrics_listed(
        ground_truth_array=masks, prediction_array=preds)
    return miou_list, pix_acc_list, F1_s_list

def metrics_list(dir_name, iteration):
    miou_list, pix_acc_list, F1_s_list = metrics_model_listed(f"{dir_name
        }/{dir_name}_{iteration}/PREDICTIONS", f"{dir_name}/{dir_name}_{
        iteration}/MASKS")
    miou_list = np.array(miou_list)
    pix_acc_list = np.array(pix_acc_list)
    F1_s_list = np.array(F1_s_list)
    return miou_list, pix_acc_list, F1_s_list

if __name__ == "__main__":
    # Seleccionando modelos a ser comparados
    dir_names = [('UNET_32', 5), ('UNET_FFT_32', 4)]
    #dir_names = [('UNET_16', 4), ('UNET_FFT_16', 3)]
```

```
    all_miou_values = []
    all_pix_acc_values = []
    all_f1_s_values = []
    for dir_name, iteration in dir_names:
        miou_list, pix_acc_list, F1_s_list = metrics_list(dir_name,
            iteration)
        all_miou_values.append(miou_list)
        all_pix_acc_values.append(pix_acc_list)
        all_f1_s_values.append(F1_s_list)
    # Convirtiendo listas a arrays de numpy
    all_miou_values = np.array(all_miou_values)
    all_pix_acc_values = np.array(all_pix_acc_values)
    all_f1_s_values = np.array(all_f1_s_values)

    statistics, p_value = wilcoxon(all_miou_values[0], all_miou_values[1])
    print( "Estadistica de prueba:" , statistics)
    print( "Valor P:" , p_value)
    print("==============================")

    n = 160
    mean_T = n * (n + 1) / 4
    std_T = np.sqrt(n * (n + 1) * (2 * n + 1) / 24)
    # Calculo del valor Z
    z_value = (statistics - mean_T) / std_T
    z_value = round(z_value, 5)
    print("El valor de z es: ", z_value)
    print("==============================")

    # Nivel de significancia
    alpha = 0.05

    # Calcular el valor critico para una prueba bilateral
    z_critical = norm.ppf(1 - alpha / 2)  # Para 0.025 en ambas colas
    print(f"Valor critico z (para alpha/2 = 0.025): + - {z_critical:.2f}")

    p_value_from_z = 2 * norm.cdf(z_value)
    print("El valor de p basado en z es:", p_value_from_z)
    print("==============================")

    # Comprobar si el estadistico Z cae en la region de rechazo
    if statistics < -z_critical or statistics > z_critical:
        print(f"Rechazamos H0: el valor z = {statistics:.4f} esta fuera de
            los limites criticos + - {z_critical:.4f}")
    else:
        print(f"No rechazamos H0: el valor z = {statistics:.4f} esta dentro
            de los limites criticos + - {z_critical:.4f}")
```

Apéndice I

Presupuesto

En la Tabla I.1 se muestra el presupuesto estimado para la realización de la presente tesis, este contempla los recursos tales como equipos, software, datos y útiles, la fuente de financiamiento y su costo estimado, así como el costo de las actividades realizadas.

Tabla I.1
Presupuesto de recursos y costo de actividades.

Categoría	**Descripción**	**Fuente de Financiamiento**	**Monto (S/.)**
Equipo	Estación de Trabajo HP Z4	LIECAR	0
Equipo	SSD Kingston 1TB	Personal	250
Software	SO Linux	Licencia libre	0
Software	Python, Tensorflow	Licencia libre	0
Datos	Imágenes satelitales	Licencia libre	0
Útiles de escritorio	Papelería en general	Personal	100
Subtotal			**350**
Actividades	**Horas estimadas**	**Tarifa por hora (S/.)**	**Monto (S/.)**
Revisión bibliográfica	40	8	320
Elaboración del Conjunto de Datos	120	8	960
Procesamiento de Datos	40	8	320
Desarrollo de los modelos	60	8	480
Entrenamiento	40	8	320
Experimentos	80	8	640
Redacción de la tesis	220	8	1 760
Subtotal	**600**		**4 800**
Total			**5 150**