FICHA RESUMEN

TÍTULO: Análisis de imágenes satelitales para la clasificación de ecosistemas en predios.

1. ÁREA DE TRABAJO: Segmentación de imágenes
2. TIPO DE PROYECTO (Aplicado, Innovación, Investigación): Aplicado
3. ESTUDIANTE(S): Francisco Santos y Roberto Quiroz
4. CORREO ELECTRÓNICO: fsantospenuela@javerianacali.edu.co y robertoegmoscarella@javerianacali.edu.co
5. DIRECCIÓN Y TELEFONO:
6. DIRECTOR: Omar Castaño
7. VINCULACIÓN DEL DIRECTOR:
8. CORREO ELECTRÓNICO DEL DIRECTOR: andreskmzw87@gmail.com
9. CO-DIRECTOR (Si aplica): No aplica
10. GRUPO O EMPRESA QUE LO AVALA (Si aplica): Fundación Cataruben
11. OTROS GRUPOS O EMPRESAS: No Aplica
12. PALABRAS CLAVE (al menos 5): Imágenes, Segmentación, Clasificación, Redes Neuronales
13. FECHA DE INICIO: 15 Enero 2024
14. DURACIÓN ESTIMADA (En meses): 10
15. RESUMEN:

El proyecto aplicado se centra en la segmentación de imágenes satelitales de predios para identificar ecosistemas. Aborda la problemática del trabajo manual requerido para segmentar zonas en imágenes, especialmente en la elaboración de proyectos de bonos de carbono. Se desarrolló un algoritmo funcional que permita a los investigadores segmentar grandes extensiones de tierra de manera eficiente, reduciendo el tiempo necesario para esta tarea. Los resultados obtenidos permiten la automatización del proceso de segmentación, particularmente sobre zonas verdes y cuerpos de agua permitiendo la evaluación de su extensión. Las posibles aplicaciones de este proyecto abarcan la investigación ambiental, la planificación del uso del suelo y la gestión de recursos naturales.

ANÁLISIS DE IMÁGENES SATELITALES PARA LA CLASIFICACIÓN DE ECOSISTEMAS EN PREDIOS

Francisco J. Santos
Roberto E. Quiroz Moscarella

Proyecto Aplicado para optar al título de
Magister en Ciencia de Datos

Director(a)
Omar Castaño

FACULTAD DE INGENIERÍA Y CIENCIAS
MAESTRÍA EN CIENCIA DE DATOS
SANTIAGO DE CALI, JUNIO 17 DE 2024

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

En un mundo cada vez más influenciado por el cambio climático y las acciones humanas, la capacidad de monitorear y comprender nuestros ecosistemas se ha convertido en una tarea esencial. Los ecosistemas, desde vastas selvas tropicales hasta frágiles humedales, desempeñan roles cruciales en la regulación climática, la biodiversidad y los servicios ecosistémicos que sustentan la vida humana. Sin embargo, su vastedad y complejidad hacen que su monitoreo sea un desafío constante.

El advenimiento de la tecnología satelital ha ofrecido una ventana sin precedentes para observar la Tierra desde el espacio, proporcionando imágenes detalladas que cubren grandes extensiones de territorio en tiempos cortos. Sin embargo, el análisis y comprensión de estas imágenes requieren una combinación de experticia en ciencias de la Tierra y herramientas tecnológicas avanzadas.

En este contexto, se ha desarrollado un modelo de aprendizaje automático basado en una arquitectura CNN U-Net que analiza imágenes satelitales para clasificar ecosistemas, específicamente diferenciando entre agua y bosques. Este proyecto combina las capacidades de observación de la tecnología satelital con las herramientas avanzadas de análisis de datos, creando un sistema robusto que permite una gestión ecológica más informada y proactiva.

El modelo, que utiliza tres backbones distintos, ofrece a investigadores, tomadores de decisiones y conservacionistas una herramienta eficaz para comprender de manera rápida y precisa el estado y la evolución de los ecosistemas, contribuyendo así al desarrollo sostenible y a la conservación a largo plazo.

1. DEFINICION DEL PROBLEMA

1.1 PLANTEAMIENTO DEL PROBLEMA.

Dentro del ámbito conocido como el "Mercado de Carbono", los "Bonos de Carbono" destacan como uno de los activos intangibles más negociados. En esencia, estos bonos representan una certificación que respalda la extracción o compensación de una tonelada de CO2 o gases de efecto invernadero de la atmósfera. La generación de estos bonos puede llevarse a cabo de diversas maneras, siendo la preservación de ecosistemas la más común para la compensación de emisiones y la captura de carbono. Este proceso implica la ejecución de un proyecto que demuestre, de principio a fin, la reducción de emisiones fruto de la preservación/restauración del predio (ecosistema) en cuestión, y el bono resultante es emitido por una entidad internacional.

Un criterio fundamental para la generación de estos bonos es la clasificación del ecosistema, ya que, en ciertos casos, puede llegar a descalificar un proyecto desde el principio. Esta clasificación debe seguir estándares específicos, utilizando información sobre los terrenos proporcionada por gobiernos o propietarios. Sin embargo, la fuente principal de datos para esta clasificación suele ser la información obtenida mediante imágenes satelitales.

A pesar de que la información proporcionada por imágenes satelitales ofrece una visión objetiva y detallada de los predios, es crucial destacar que su análisis implica una significativa inversión de capital humano. Este proceso se convierte en una tarea extensa y detallada, requiriendo la dedicación de expertos para interpretar y extraer datos significativos. La necesidad de examinar extensas áreas geográficas y evaluar diversos elementos del ecosistema contribuye a la complejidad del análisis satelital. Esta complejidad, a su vez, puede generar un cuello de botella para el avance eficiente de los proyectos, ya que la interpretación precisa de las imágenes demanda tiempo y experiencia especializada.

En naciones donde la geografía presenta una mayor constancia, la clasificación de ecosistemas para la generación de Bonos de Carbono se beneficia de la disponibilidad de información confiable por parte del gobierno. Este tipo de áreas suele tener una menor diversidad de variables ambientales, facilitando la tarea de clasificación. Sin embargo, la situación difiere en países como Colombia, caracterizado por una geografía rica y diversa.

Aquí, un solo predio puede albergar una variedad considerable de ecosistemas, dependiendo de su extensión. Este factor amplifica el desafío de clasificación, ya que la evaluación precisa de múltiples variables ambientales se vuelve más compleja. Esta complejidad puede afectar la cantidad de Bonos de Carbono que las entidades internacionales están dispuestas a conceder a proyectos en estas regiones, o incluso determinar la aceptación misma de dichos proyectos.

En el contexto colombiano, la rica diversidad geográfica agrega una capa adicional de complejidad al proceso de generación de Bonos de Carbono. La necesidad de clasificar múltiples ecosistemas un consumo considerable de tiempo. Este tiempo adicional se destina principalmente a asegurar la viabilidad del proyecto en términos de conservación y, a menudo, deja en segundo plano otros análisis cruciales. La extensa tarea de evaluar la idoneidad de un predio para la compensación de emisiones a veces puede dilatar el inicio real del proyecto, lo cual podría llevar a demoras innecesarias en la implementación de medidas efectivas de conservación.

Ante este desafío, surge la necesidad imperante de adoptar soluciones computacionales para optimizar el proceso de generación de Bonos de Carbono. Una estrategia clave sería la implementación de herramientas de análisis de datos y procesamiento de imágenes. Estas soluciones proporcionarían a los profesionales involucrados clasificaciones precisas de los diversos ecosistemas presentes en las imágenes satelitales. Al reducir significativamente los tiempos requeridos para esta fase de evaluación, se abriría la puerta a una toma de decisiones más ágil y eficiente en relación con la viabilidad de los proyectos de conservación. La integración de tecnologías computacionales se presenta como un elemento clave para enfrentar los desafíos actuales y agilizar la contribución efectiva a la mitigación del cambio climático.

1.2 FORMULACION DEL PROBLEMA

- ¿De qué manera el proyecto mejora la capacidad de monitorear cambios en los ecosistemas a gran escala?
- ¿Qué cambios significativos en los ecosistemas pueden ser detectados gracias al modelo que no se habrían identificado de otro modo?
- ¿Qué nivel de precisión alcanzó el modelo de aprendizaje automático en la clasificación de ecosistemas?
- ¿Cómo se podría adaptar el modelo en respuesta a los resultados obtenidos y los avances tecnológicos?

2. OBJETIVOS DEL PROYECTO

2.1 OBJETIVO GENERAL

Desarrollar un modelo de aprendizaje automático capaz de procesar imágenes satelitales para clasificar ecosistemas y predecir sus tendencias de vegetación.

2.2 OBJETIVOS ESPECÍFICOS

- Adquirir un conjunto de imágenes satelitales que abarquen una variedad de ecosistemas y Filtrar y retener las imágenes relacionadas a ecosistemas que sean valiosos o califiquen dentro de los estándares.
- Implementar y entrenar un modelo de aprendizaje automático para identificar y clasificar diferentes tipos de ecosistemas a partir de imágenes satelitales. Esto, de forma genérica, sabana, llanura, Etc.
- Crear un algoritmo que asigne etiquetas a la vegetación que se pueda apreciar en el ecosistema. Es decir, que logre etiquetar distintas vegetaciones presentes en el predio, más allá del tipo de ecosistema, etiquetar los bosques, cultivos, malezas Etc.
- Crear un algoritmo que pueda anticipar tendencias en la vegetación (crecimiento o decrecimiento) basado en las características identificadas en las imágenes satelitales y en la información dada por los propietarios (como que periódicamente realizan quemas).

3. MARCO DE REFERENCIA.

3.1 MARCO TEÓRICO

A continuación, se presentan los conceptos relacionados al desarrollo del proyecto, El proyecto se centra en la aplicación de técnicas avanzadas de análisis de imágenes satelitales y machine learning para la clasificación y monitoreo de ecosistemas en Colombia, con el objetivo de clasificar áreas. Este marco teórico detalla los componentes y consideraciones clave del proyecto.

3.1.2 Ecosistemas

En el sentido geográfico, un ecosistema se refiere a un área geográfica específica que comparte ciertas características climáticas, geológicas y biológicas. Estas áreas son unidades funcionales que involucran a organismos vivos y su entorno físico interactuando de manera dinámica. La geografía de un ecosistema puede variar desde pequeñas regiones, como un estanque o bosque, hasta extensiones más amplias, como una selva tropical o un desierto.

Por otra parte, Jorgenson [11] Define un ecosistema como una unidad ecológica que incluye tanto los organismos vivos (biocenosis) como su entorno físico (biotopo). los ecosistemas no solo se definen por sus características físicas, sino también por las interacciones complejas entre los organismos que habitan en ellos y su entorno. Estos sistemas incluyen tanto componentes bióticos, como plantas, animales y microorganismos, como componentes abióticos, que abarcan el suelo, el agua y el clima. Además, la geografía de un ecosistema está profundamente interconectada con factores humanos, como la urbanización y la agricultura, lo que puede alterar su equilibrio y biodiversidad.

3.1.3 Imágenes Satelitales

C. Ünsalan y K. L. Boyer [5] definen una imagen satelital como una representación de la superficie terrestre capturada por sensores a bordo de satélites, que registran diferentes longitudes de onda de la luz, permitiendo la creación de imágenes multiespectrales. Gracias a la órbita de los satélites, estas imágenes ofrecen una perspectiva única y global, lo que las convierte en herramientas poderosas para el monitoreo ambiental, la planificación urbana, la agricultura, la gestión de desastres y diversas aplicaciones científicas. La capacidad de obtener información detallada sobre grandes extensiones de terreno a lo largo del tiempo hace que las imágenes satelitales sean fundamentales para el estudio y la comprensión de procesos terrestres y cambios en el medio ambiente a escala global.

Además de su utilidad en diversas aplicaciones, las imágenes satelitales también desempeñan un papel crucial en la investigación y el monitoreo de fenómenos naturales y cambios climáticos. Permiten a los científicos observar patrones de uso del suelo, cambios en la cobertura forestal y el impacto de eventos extremos, como incendios forestales o inundaciones.

Figura 1: Imagen satelital, cuerpo de agua.

En el contexto de este trabajo, las imágenes que serán tratadas provienen de plataformas como Google Earth, así como de satélites comerciales de pago que ofrecen imágenes de alta resolución. Estas imágenes suelen ser preprocesadas por la fundación lo que permite un nivel de detalle considerablemente alto, útil para diversas aplicaciones que requieren precisión en la visualización de la superficie terrestre.

Una vez obtenidas las imágenes, se realiza un procesamiento adicional para optimizar su uso según los objetivos del análisis. Este proceso incluye la mezcla de bandas, un método mediante el cual se combinan diferentes longitudes de onda para resaltar características específicas de la superficie. Dependiendo del tipo de superficie que se desee estudiar, se eligen combinaciones de bandas espectrales específicas. Por ejemplo, para identificar vegetación densa o áreas forestales, se utilizarían combinaciones que incluyan el infrarrojo cercano y el rojo.

3.1.4 El procesamiento de imágenes

El procesamiento de imágenes es definido por T. Acharya y A. K. Ray [6] como como una etapa crucial en el procesamiento de imágenes que se lleva a cabo antes de la aplicación de técnicas más complejas. El objetivo del preprocesamiento es mejorar la calidad de las imágenes y preparar los datos para un análisis posterior. Es un campo multidisciplinario que combina técnicas de matemáticas, informática y visualización para transformar y analizar imágenes. Este proceso no solo se limita a mejorar la calidad visual de una imagen, sino que también busca facilitar la extracción de información relevante para diferentes aplicaciones. Es un conjunto de técnicas y

métodos utilizados para manipular, analizar y mejorar imágenes digitales. Este campo abarca una amplia variedad de operaciones que se aplican a las imágenes con el objetivo de extraer información útil.

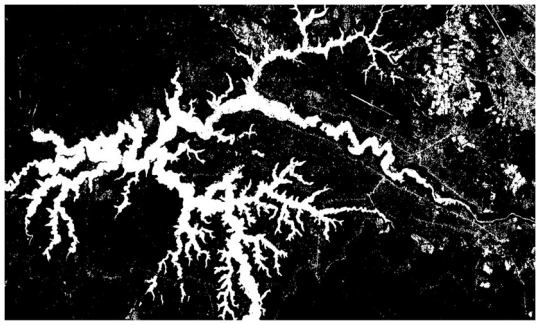

Figura 2: Ejemplo de segmentación

De forma más generalizada, los autores [6] proponen la siguiente metodología para el análisis y procesamiento de imágenes

Adquisición de Imágenes: Captura de imágenes utilizando diferentes tipos de sensores y dispositivos.

Corrección de Iluminación: Ajuste de la iluminación y la exposición para reducir sombras y mejorar la uniformidad.

Filtrado: Aplicación de filtros (por ejemplo, de suavizado o de realce) para eliminar ruido y mejorar características específicas de la imagen.

Ajuste de Contraste y Brillo: Modificación de la intensidad de píxeles para mejorar la visibilidad de los detalles.

Transformaciones Geométricas: Ajustes en la geometría de la imagen, como rotación, escalado y recorte.

Segmentación: Dividir la imagen en partes significativas o regiones para facilitar el análisis posterior.

3.1.6 Modelos de clasificación

Los autores D. Michie, D. J. Spiegelhalter, y C. C. Taylor [8] entienden un modelo de clasificación como un método que asigna etiquetas o categorías a datos basándose en características observadas. La clasificación se utiliza para predecir la categoría a la que pertenece una nueva observación, utilizando información aprendida de un conjunto de datos previamente etiquetado. De forma generalizada los autores asignan a un modelo las siguientes características:

Entrenamiento: Un modelo se entrena utilizando un conjunto de datos que contiene ejemplos con sus respectivas etiquetas. Durante esta fase, el modelo aprende las relaciones entre las

características y las categorías.
Generalización: El objetivo es que el modelo no solo memorice los datos de entrenamiento, sino que también generalice bien a datos no vistos, permitiendo hacer predicciones precisas.
Tipos de Modelos: Existen diferentes enfoques para la clasificación, incluyendo métodos estadísticos (como regresión logística), algoritmos de aprendizaje automático (como máquinas de soporte vectorial), y redes neuronales.
xEvaluación: La efectividad del modelo se evalúa mediante métricas como precisión, sensibilidad, especificidad y la matriz de confusión, entre otras.

3.1.7 Redes Neuronales

Gerón [1] como un modelo computacional inspirado en la forma en que funciona el cerebro humano. Consiste en una serie de capas de neuronas artificiales que procesan datos y aprenden a realizar tareas específicas a través del entrenamiento. El autor propone las siguientes características para una red neuronal cualquiera:

Estructura: Una red neuronal típica consta de una capa de entrada, varias capas ocultas y una capa de salida. Cada capa está formada por neuronas que reciben entradas, las procesan y producen salidas.

Función de Activación: Cada neurona utiliza una función de activación (como ReLU o sigmoide) para determinar si debe "disparar" (activar) en función de la suma ponderada de sus entradas.

Entrenamiento: Las redes neuronales se entrenan mediante algoritmos de optimización, como el descenso por gradiente, ajustando los pesos de las conexiones para minimizar el error entre las predicciones y las etiquetas reales.

Géron [1] menciona varias aplicaciones de redes neuronales, incluyendo:

1. **Reconocimiento de Imágenes**: Clasificación de imágenes y detección de objetos en fotografías o videos.
2. **Procesamiento de Lenguaje Natural**: Tareas como traducción automática, análisis de sentimientos y generación de texto.
3. **Juegos**: Entrenamiento de modelos para jugar y competir en juegos complejos, como ajedrez o Go.
4. **Predicción de Series Temporales**: Modelos que predicen valores futuros basándose en datos históricos.

5. **Diagnóstico Médico**: Análisis de imágenes médicas y datos de pacientes para ayudar en diagnósticos y tratamientos.

3.1.8 Redes Neuronales Convolucionales

En "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" [1], Aurélien Géron explica que una red neuronal convolucional (CNN, por sus siglas en inglés) es un tipo de red neuronal especialmente diseñada para procesar datos que tienen una estructura de cuadrícula, como imágenes. Destaca que las CNNs son ampliamente utilizadas en tareas de visión por computadora debido a su capacidad para capturar patrones espaciales en datos de imagen.

Géron explica que las CNNs utilizan capas convolucionales para aplicar filtros a regiones locales de la entrada de la imagen, lo que permite detectar características importantes, como bordes, texturas o patrones, en diferentes partes de la imagen. Además, utiliza capas de agrupación (pooling) para reducir la dimensionalidad de la salida de las capas convolucionales, lo que ayuda a extraer características relevantes y a mejorar la eficiencia computacional.

En resumen, Aurélien Géron describe las CNNs como un tipo de red neuronal especializada en procesar datos de imagen, que utiliza capas convolucionales y de agrupación para capturar y procesar eficientemente patrones espaciales en imágenes.

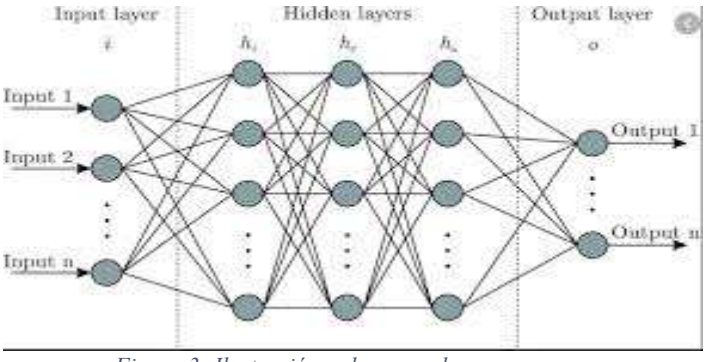

Figura 3: Ilustración red neuronal

Preprocesamiento: Antes de ser alimentadas a la red, las imágenes pueden ser preprocesadas. Esto incluye pasos como la normalización (ajustar los valores de píxeles a un rango entre 0 y 1), el redimensionamiento (ajustar las dimensiones a las requeridas por la red) y, a veces, la augmentación de datos (como rotaciones o cambios de brillo) para aumentar la diversidad del conjunto de entrenamiento.

Capa de Entrada: Una vez preprocesadas, las imágenes se introducen en la capa de entrada de la CNN. Esta capa recibe el volumen de datos y lo prepara para ser procesado por las capas convolucionales.

Capa de Salida: Al final de la red, la CNN tiene una o más capas densas que generan la salida. La última capa de la red es generalmente una capa densa que utiliza una función de activación como softmax (en el caso de clasificación múltiple) o sigmoide (en el caso de clasificación binaria) para convertir las características extraídas en probabilidades de clase.

En este trabajo se va a usar una red neuronal tipo U-Net es una arquitectura especialmente diseñada para tareas de segmentación de imágenes, como la segmentación semántica en imágenes médicas o la segmentación de objetos en imágenes. Su principal característica es su estructura de codificador-decodificador simétrica, en la que el codificador (parte descendente de la red) reduce progresivamente la resolución de la imagen, extrayendo características a medida que la imagen se va comprimiendo, mientras que el decodificador (parte ascendente de la red) reconstruye la imagen original con la resolución deseada, a partir de las características extraídas. Lo que distingue a U-Net de otras redes neuronales es la conexión directa entre el codificador y el decodificador, que se logra mediante saltos de conexión (skip connections). Estas conexiones permiten que la red mantenga detalles importantes de la imagen a través de diferentes niveles de resolución, mejorando la precisión de la segmentación.

El diseño de U-Net fue inicialmente propuesto para la segmentación de imágenes biomédicas, pero su flexibilidad ha permitido su adopción en una amplia gama de tareas de segmentación. La arquitectura básica consta de una serie de bloques convolucionales seguidos de capas de pooling para reducir la dimensión espacial de la imagen. Después, una vez que la imagen ha sido comprimida por el codificador, se pasa a través de una serie de capas de transposición convolucional (también conocidas como "upsampling") en el decodificador para aumentar la resolución. Lo innovador de U-Net es la forma en que las características extraídas en el codificador

se reutilizan mediante las conexiones de salto, lo que permite a la red recuperar detalles finos que de otro modo se perderían durante el proceso de reducción y ampliación de la imagen.

La arquitectura U-Net ha demostrado ser excepcionalmente efectiva en tareas de segmentación, principalmente debido a su capacidad para aprender de manera eficiente tanto las características globales de una imagen como los detalles más precisos. Además de las conexiones de salto, otra de sus características clave es el uso de convoluciones de tamaño pequeño (por lo general, 3x3), lo que permite que la red tenga un número relativamente bajo de parámetros en comparación con otras arquitecturas profundas. Esto, combinado con la presencia de regularización y técnicas de optimización como la normalización de lotes (batch normalization), hace que U-Net sea capaz de generalizar bien a nuevas imágenes, incluso con un número limitado de datos de entrenamiento.

una parte fundamental del proceso será experimentar con diferentes backbones para determinar cuál proporciona el mejor rendimiento para el conjunto de datos específico. Un backbone, en el contexto de redes neuronales, es la parte de la red que se encarga de la extracción de características a partir de las imágenes de entrada. Generalmente, el backbone está formado por capas convolucionales que identifican patrones y características importantes de las imágenes, como bordes, texturas o formas complejas, antes de que estas características sean procesadas por las capas superiores para realizar tareas específicas, como la segmentación o la clasificación. El backbone es esencial, ya que proporciona la representación inicial y fundamental de los datos que la red usará para hacer predicciones.

En este caso, se probarán diferentes arquitecturas de backbones preentrenados, como ResNet, VGG o EfficientNet, que son modelos bien establecidos y entrenados previamente en grandes conjuntos de datos como ImageNet. Estos modelos han demostrado ser efectivos para la extracción de características en tareas de visión computacional y, por lo tanto, al integrarlos en la red U-Net, se espera mejorar la capacidad de la red para segmentar imágenes con mayor precisión y eficiencia. Cada backbone tiene una estructura distinta, lo que significa que cada uno puede extraer diferentes tipos de características de las imágenes de entrada, lo que puede influir en la calidad de la segmentación producida por la red U-Net. Así, el objetivo será evaluar cuál de estos backbones proporciona la mejor base para las tareas de segmentación en el contexto de este trabajo.

El proceso de experimentación con diferentes backbones en la red U-Net incluirá la evaluación de su rendimiento en términos de precisión, velocidad de entrenamiento y capacidad de generalización en nuevos datos. Cada backbone será evaluado a través de técnicas como la validación cruzada, asegurando que se identifique el modelo que mejor se adapte a las necesidades específicas del proyecto. Este enfoque de experimentar con diferentes backbones es

crucial, ya que, aunque todos los backbones están diseñados para extraer características, algunos pueden ser más adecuados que otros dependiendo de la naturaleza de las imágenes y los objetivos de segmentación. De esta manera, se podrá seleccionar el backbone más eficiente y eficaz para la tarea en cuestión.

3.1.9 Aprendizaje Profundo

Goodfellow, Y. Bengio, y A. Courville [12] definen el aprendizaje profundo como una subárea del aprendizaje automático que utiliza redes neuronales profundas para modelar datos. Estas redes se caracterizan por tener múltiples capas de neuronas, lo que les permite aprender representaciones jerárquicas y complejas de los datos. Los autores resaltan una serie de características importantes de este tipo de aprendizaje.

Arquitectura Profunda: Consiste en múltiples capas de procesamiento, donde cada capa transforma la representación de los datos de una manera que puede capturar características cada vez más abstractas.

Autoaprendizaje: Las redes profundas pueden aprender automáticamente características relevantes de los datos sin necesidad de ingeniería de características manual, lo que las hace muy eficaces en tareas complejas.

Uso de Grandes Conjuntos de Datos: El aprendizaje profundo se beneficia significativamente de grandes volúmenes de datos y potentes recursos computacionales, lo que permite entrenar modelos más precisos.

Algoritmos de Optimización: Se utilizan técnicas como el descenso por gradiente estocástico y variaciones de este para ajustar los pesos de las redes durante el entrenamiento.

3.2 ANTECEDENTES

Se encontraron varios antecedentes donde se realiza, precisamente, un modelo de aprendizaje (tipo clasificación) para el procesamiento de imágenes satelitales con el fin de generar una serie de clasificaciones.

En el primer antecedente H. Ferdous, T. Siraj, S. J. Setu, M. M. Anwar, y M. A. Rahman [2] realizan un modelo con el fin de abordar y clasificar distintos tipos de imágenes satelitales en general. En el segundo, se realiza un mapeo de un lago a través del procesamiento de imágenes satelitales y finalmente, se encontró un antecedente donde se busca generar una clasificación para ciertas zonas presentes en los ecosistemas de tipo manglar. En general, este tema ha sido abordado desde una perspectiva muy general hasta un enfoque particular relacionado precisamente a la clasificación de ecosistemas.

La investigación se centra en la implementación de algoritmos sofisticados y métodos avanzados para resolver problemas de clasificación, especialmente en el contexto de redes y sensores satelitales avanzados. Se destaca la necesidad de utilizar métodos de aprendizaje automático supervisado, como el vecino más cercano (KNN), redes neuronales artificiales (ANN) y árboles de decisión, optimizados para trabajar eficientemente con sistemas de procesamiento gráfico (GPU). El estudio preliminar, basado en experimentos extensivos con conjuntos de datos de imágenes satelitales, sugiere que el sistema de clasificación propuesto puede representar una alternativa competitiva en términos de precisión y eficacia frente a los esquemas actuales basados en extracción de características.

En el segundo antecente encontrado a través del análisis de imágenes de satélite Landsat que abarcan tres décadas y utilizando Google Earth Engine M. K. Kolli, C. Opp, D. Karthe, y M. Groll [3], realizan un estudio que identifica y clasifica escenarios de uso del suelo para diferentes años: pre-restauración (1999), inmediatamente después de la restauración (2008) y situación actual de un lago en India (2018). Se utilizan índices clave como el NDVI (Índice de Vegetación de Diferencia Normalizada) y el NDWI (Índice de Agua de Diferencia Normalizada) para seguir las dinámicas de cobertura terrestre, revelando esfuerzos exitosos de restauración que devolvieron el lago a su estado anterior, como en 1999.

Como tercer antecedente N. B. Toosi, A. R. Soffianian, S. Fakheran, S. Pourmanafi, C. Ginzler, y L. T. Waser [4] Investigam la aplicación de técnicas avanzadas de teledetección para mapear y clasificar ocho clases de cobertura terrestre en un área de 768 km2. Los manglares son ecosistemas altamente productivos que ofrecen numerosos bienes y servicios ecológicos y económicos. El estudio emplea un enfoque de ampliación (upscaling) que incluye la extracción de valores de reflectancia de imágenes Worldview-2, la segmentación basada en características espectrales y espaciales, y la predicción de cobertura terrestre basada en imágenes Sentinel-2. Esta metodología, que minimiza costos al usar datos satelitales gratuitos y comerciales, logró una precisión general del 65.5% y un coeficiente kappa de 0.63

4. DESARROLLO DE LOS OBJETIVOS

La fundación proporcionó para este proyecto un conjunto de datos compuesto por carpetas con un total de 2481 imágenes satelitales de cuerpos de agua y 1453 imágenes de bosques, todas con sus respectivas máscaras de segmentación. Estas imágenes, de alta resolución, fueron fundamentales para el análisis y la segmentación. Además, la fundación entregó una serie de imágenes en formato TIFF que habían sido preprocesadas por ingenieros agroforestales, en las cuales las combinaciones de bandas eran de una mayor complejidad, orientadas a proporcionar información más detallada sobre las superficies. Sin embargo, estas últimas imágenes, debido a su extenso procesamiento, fueron descartadas, ya que presentaban un tratamiento tan avanzado que las imágenes ya se encontraban pseudo-segmentadas, lo que interfería con el propósito original del proyecto, que era realizar una segmentación desde cero. Por lo tanto, se decidió trabajar con las imágenes que se encontraban en un formato "estándar", con un preprocesamiento mínimo, lo que permitió realizar una segmentación más precisa y adecuada a los objetivos del proyecto.

Se llevó a cabo un análisis exhaustivo de los entregables proporcionados por la empresa. Este análisis incluyó una serie de pasos cuidadosamente planificados y ejecutados, con el propósito de filtrar y organizar las imágenes satelitales que contenían áreas de bosques y cuerpos de agua, junto con sus máscaras correspondientes. A continuación, se detallan las acciones realizadas:

1. **Conteo de archivos por carpeta:** Inicialmente, se implementó un procedimiento para contar los archivos presentes en cada subcarpeta dentro del directorio principal donde se almacenaban las imágenes y máscaras recibidas de la empresa. El objetivo de esta etapa

fue evaluar la cantidad y distribución de archivos disponibles. Para ello, se recorrieron todas las carpetas, incluyendo aquellas que pudieran estar vacías. Durante el recorrido, se excluyeron las carpetas que contenían únicamente subdirectorios, centrándonos en aquellas que tenían archivos útiles. Los resultados obtenidos fueron representados mediante un gráfico de barras horizontal, lo que permitió visualizar de manera clara la cantidad de archivos por carpeta y detectar posibles inconsistencias en la distribución de datos.

2. **Validación de coincidencias entre imágenes y máscaras:** Una vez obtenido el conteo de archivos, se procedió a la validación de la correspondencia entre las imágenes y las máscaras, tanto en las carpetas principales como en sus subcarpetas. Para ello, se extrajeron los nombres de los archivos en ambas carpetas (imágenes y máscaras) eliminando sus extensiones, lo que permitió realizar una comparación precisa. Se verificó que cada subcarpeta de imágenes tuviera una subcarpeta equivalente en las máscaras y que los nombres de archivos coincidieran. Se generó un informe detallado de los archivos y carpetas que no tenían pareja, especificando su ubicación y tipo de error. Estos resultados se visualizaron en un gráfico de barras, diferenciando claramente entre la cantidad de archivos coincidentes y los que carecían de una contraparte, facilitando la detección de inconsistencias en la organización de los datos entregados por la empresa.

Cuerpos de agua:

Figura 4: cantidad de imágenes y máscaras agua

Cuerpos de bosques:

Figura 5 Cantidad de imágenes y máscaras Bosque

3. **Filtrado y distribución de imágenes en conjuntos de datos:** Con las imágenes y máscaras validadas, se procedió a organizar los datos para el proceso de segmentación. Para esto, se establecieron tres conjuntos de datos: entrenamiento, validación y prueba. Antes de iniciar la distribución, se crearon las carpetas necesarias para cada conjunto, asegurando que la estructura de directorios cumpliera con los estándares requeridos. A continuación, se seleccionaron las imágenes y las máscaras, emparejándolas cuidadosamente. Se realizó una verificación para garantizar que el número de imágenes coincidiera con el de las máscaras. Luego, se procedió a seleccionar aleatoriamente las parejas de imágenes y máscaras para asegurar la aleatoriedad en los conjuntos de datos. Se definieron proporciones de distribución: 70% de los datos para el entrenamiento, 20% para la validación y 10% para la prueba. Los archivos fueron copiados a las carpetas correspondientes, manejando adecuadamente posibles errores durante el proceso de copia, como conflictos de nombres o problemas de acceso a los archivos. Finalmente, se generó un resumen indicando la cantidad de archivos copiados a cada conjunto.

Directorio	***Cantidad de Archivos***
Aguas\test	285
Aguas\test_labels	285
Aguas\train	1988
Aguas\train_labels	1988
Aguas\val	568

Aguas\val_labels	568
Total imagenes	**5682**

Tabla 1: Imágenes procesadas - Aguas

Directorio	**Cantidad de Archivos**
Bosques\test	146
Bosques\test_labels	146
Bosques\train	1017
Bosques\train_labels	1017
Bosques\val	290
Bosques\val_labels	290
Total imagenes	**2906**

Tabla 2: Imágenes procesadas - Bosques

Como resultado de este análisis y organización, se logró obtener un conjunto de imágenes y máscaras que cumplían con los estándares requeridos por la empresa. Las imágenes fueron filtradas, validadas y distribuidas en conjuntos de datos claramente definidos, listos para las siguientes etapas del proceso de segmentación. Este procedimiento garantizó la calidad y organización necesaria para continuar con el desarrollo del proyecto de segmentación.

A continuación, se presenta el flujograma del presente objetivo:

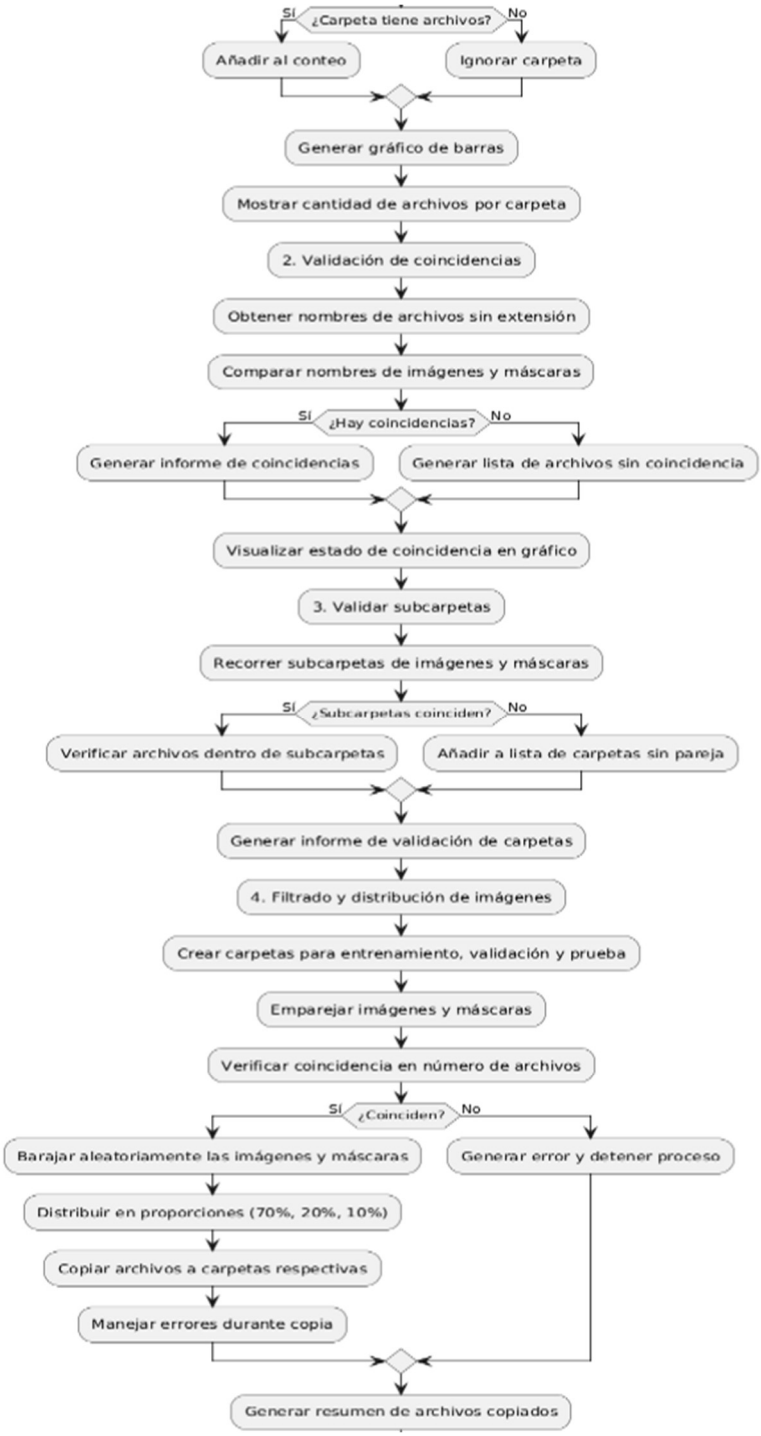

Figura 6: Ejemplo de imágenes a clasificar

1. CLASIFICACIÓN DE IMÁGENES

Se creó una **Red Neuronal Convolucional (CNN)**. Este tipo de red es especialmente eficaz para tareas de clasificación de imágenes y reconocimiento de patrones visuales.

La arquitectura de la red neuronal presentada es una **Red Neuronal Convolucional (CNN)** compuesta por varias capas que procesan las imágenes de entrada. Comienza con una capa de entrada que define las dimensiones de las imágenes (altura, ancho y 3 canales de color). A continuación, se aplica una normalización de los píxeles mediante la capa de escalado, que transforma los valores de píxeles de 0-255 a un rango de 0 a 1. Luego, la red incluye tres bloques de capas convolucionales, cada uno seguido de una capa de MaxPooling. Estas capas convolucionales extraen características locales y patrones de la imagen, mientras que las capas de pooling reducen la dimensionalidad y retienen las características más importantes, mejorando la eficiencia y robustez del modelo.

Al final de la arquitectura, las características extraídas se aplanan mediante una capa Flatten, convirtiendo los mapas de características en un vector unidimensional. Este vector se alimenta a dos capas densas: la primera con 128 neuronas, que permite una mezcla de características complejas, y la segunda, que tiene un número de neuronas equivalente a la cantidad de clases que se desean predecir (amount). La última capa aplica una función de activación adecuada para la clasificación, permitiendo que la red produzca las probabilidades de pertenencia a cada clase, lo que facilita la tarea de clasificación de imágenes.

Esta es una tarea relativamente simple, se logró que la red fuese capaz de reconocer patrones con una precisión alta y por lo tanto predecir que clase pertenece la imagen que se le entrega (desierto, agua, bosque etc).

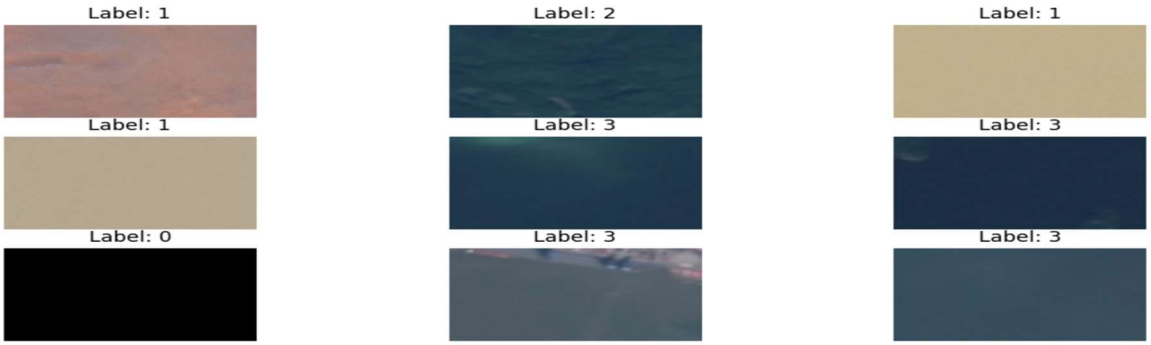

Figura 7: Ejemplo de imágenes a clasificar

Después de 141 épocas la red era capaz de predecir de forma satisfactoria la imagen que se le fuese presentada.

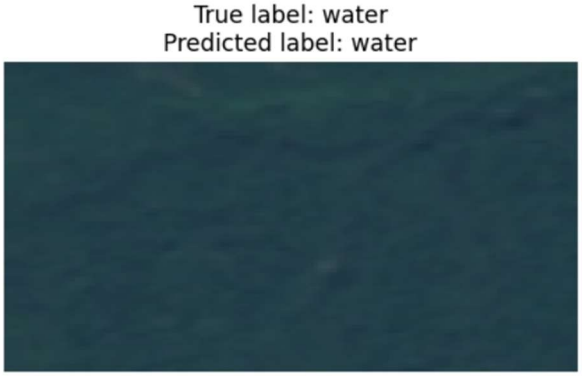

Figura 8: Ejemplo clasificación

Este es un paso crucial para la fundación al proponer proyectos relacionados con bonos de carbono. Antes de proceder con la segmentación, es necesario identificar y verificar la extensión de las áreas de bosques o cuerpos de agua que serán tratadas, ya que el proyecto debe ser auditado y validado. Para ello, se requiere que los ingenieros comprueben, a través de imágenes satelitales, que las áreas propuestas realmente existen en el terreno, lo cual implica una inspección visual detallada de la zona. Este proceso de clasificación previo es esencial, ya que permite que el modelo identifique de manera efectiva, a través de diferencias de color, si una región es un área verde (bosque) o no, lo cual facilita la validación técnica. Tras la clasificación, los ingenieros pueden realizar una comprobación adicional en función de los estándares técnicos relacionados con la flora, aunque la existencia de estas áreas debe ser confirmada previamente a través del análisis visual. Así, la clasificación se convierte en un paso crucial para asegurar la

veracidad del proyecto antes de proceder con el análisis y segmentación detallada.

5. SEGMENTACIÓN DE IMÁGENES

Durante el desarrollo de nuestro proyecto, decidimos entrenar modelos separados para la segmentación de agua y bosques, dado que el objetivo era segmentar cada tipo de clase de manera independiente.

Esta decisión se alineó con la forma en que la empresa maneja sus procesos, ya que el análisis y la segmentación de los datos de agua y bosques se realizan de forma separada para obtener resultados más precisos y específicos en cada área.

La empresa gestiona el análisis de estos ecosistemas de manera independiente debido a las características particulares de cada uno, que requieren enfoques especializados. Por ejemplo, las imágenes relacionadas con cuerpos de agua tienen patrones y texturas muy diferentes en comparación con las zonas boscosas, lo que significa que un solo modelo podría tener dificultades para capturar y segmentar de manera eficaz ambos tipos de ecosistemas. Al segmentar por separado:

Cada modelo se enfocó en aprender características únicas de su clase (agua o bosques), maximizando la precisión de los resultados al estar adaptado específicamente para el contexto de cada ecosistema.

Los procesos operativos de la empresa ya están diseñados para manejar datos de manera independiente, ya que las decisiones y acciones que se derivan del análisis de cada clase son diferentes. Esto refuerza la necesidad de tener modelos que estén alineados con los flujos de trabajo y objetivos operacionales

La optimización de los modelos para cada clase permitió un mayor control sobre los resultados, facilitando la implementación de medidas correctivas y de análisis en cada segmento sin interferir con el análisis del otro ecosistema

Adicionalmente, decidimos enfocarnos únicamente en la arquitectura U-Net con backbones preentrenados (ResNet, EfficientNet y VGG). Esta decisión se basó en la flexibilidad y eficiencia que ofrece U-Net para tareas de segmentación de imágenes con máscaras binarias, lo que la hizo ideal para los datasets de agua y bosques que estábamos manejando.

Al utilizar backbones preentrenados en ImageNet, como ResNet, EfficientNet y VGG, pudimos aprovechar el conocimiento transferido de modelos ya entrenados en un gran conjunto de

imágenes generales, mejorando así la capacidad de nuestro modelo para extraer características relevantes desde las imágenes satelitales que estábamos segmentando.

ResNet: Seleccionamos ResNet34 y ResNet50 como backbones por su capacidad de profundizar en la extracción de características, manteniendo un buen equilibrio entre rendimiento y costo computacional. Nos permitió capturar de manera eficiente los detalles en las imágenes, tanto en agua como en bosques.

EfficientNet: También implementamos EfficientNet, un modelo conocido por ofrecer alta precisión con menor uso de recursos computacionales. Su capacidad de escalabilidad lo hizo una opción excelente para manejar datasets grandes sin comprometer el rendimiento.

VGG16: A pesar de ser una arquitectura más simple, VGG16 resultó ser una opción robusta, especialmente cuando el tamaño del dataset no era tan grande. Su simplicidad nos permitió obtener resultados consistentes sin añadir demasiada complejidad.

Durante el desarrollo del modelo U-Net para la segmentación de agua, teníamos un total de 2841 imágenes originales y el mismo número de máscaras correspondientes. Para estructurar correctamente el dataset, decidimos dividirlo en tres conjuntos: entrenamiento, validación y prueba. Aplicamos una división común de 70% para entrenamiento, 20% para validación y 10% para prueba.

Esta distribución nos permitió entrenar, validar y evaluar el modelo de manera balanceada, asegurándonos de que el conjunto de prueba fuera representativo del rendimiento general del modelo y que la validación sirviera para ajustar los hiperparámetros durante el proceso de entrenamiento.

Comenzamos el proceso fijando una semilla para los generadores de números aleatorios en Python, NumPy y TensorFlow, garantizando así la reproducibilidad de nuestros experimentos. A continuación, procedimos a cargar las imágenes satelitales y sus respectivas máscaras desde los directorios designados, utilizando un generador de datos personalizado. Este generador se encargó de cargar las imágenes y las máscaras en lotes, facilitando la gestión de la memoria durante el entrenamiento del modelo. Para asegurar la correcta carga de los datos, realizamos una visualización de las primeras imágenes junto con sus máscaras, verificando que estuvieran correctamente alineadas.

- **Primeras imágenes de entrenamiento:**

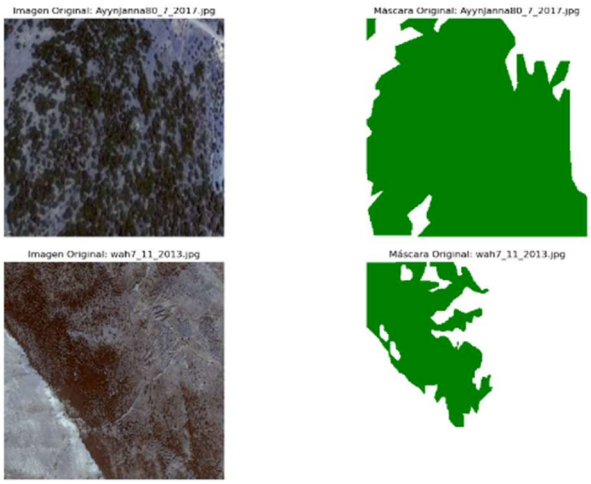

Figura 9: Ejemplo Imágenes obtenidas y máscaras.

Durante esta etapa, creamos un proceso específico para la carga de las máscaras en formato binario (blanco y verde). Dado que nuestro objetivo era clasificar dos clases (bosques y agua), necesitábamos que las máscaras tuvieran únicamente dos valores para simplificar la tarea de segmentación. Para lograr esto, implementamos un método que cargaba las máscaras en color y luego identificaba los píxeles verdes y blancos. Los píxeles verdes representaban las áreas de bosque y fueron asignados un valor binario de 1, mientras que los píxeles blancos, que representaban otras áreas, se asignaron un valor binario de 0. Esto permitió convertir las máscaras a un formato binario adecuado para el entrenamiento del modelo.

Elegimos esta estrategia para facilitar la tarea de segmentación del modelo y enfocar el aprendizaje en una clasificación clara entre las **áreas de interés (bosques)** y el **resto del entorno**. Además, el uso de un formato binario simplificó la aplicación de las funciones de pérdida y las métricas de evaluación, al ser una tarea de clasificación binaria en lugar de multiclase.

Selección e implementación del modelo

Para abordar el problema de clasificación, seleccionamos el modelo U-Net, con pesos iniciales de ImageNet, lo que ayudó a acelerar el entrenamiento y mejorar el rendimiento del modelo. Además, agregamos una capa de dropout en el decodificador del U-Net para reducir el riesgo de sobreajuste. La tasa de dropout se configuró como un hiperparámetro a optimizar para encontrar el equilibrio adecuado entre regularización y capacidad de generalización.

Optimización de hiperparámetros

La optimización de los hiperparámetros fue fundamental para mejorar el rendimiento del modelo. Utilizamos la herramienta Optuna para llevar a cabo la búsqueda de la mejor combinación de hiperparámetros a través de múltiples ensayos (trials). Los hiperparámetros que optimizamos incluyeron:

- **Tasa de aprendizaje:** Exploramos valores en el rango [1e-5, 1e-3] en escala logarítmica. Optuna seleccionó valores como 1e-4 y 5e-5, los cuales contribuyeron a una convergencia más precisa del modelo.
- **Backbone:** Optimizamos la elección del backbone entre ResNet34, EfficientNetB3 y VGG16, evaluando su impacto en el desempeño general del modelo.
- **Data Augmentation:** Evaluamos si aplicar o no técnicas de data augmentation, con valores posibles de True o False. La inclusión de data augmentation en algunos ensayos mejoró la capacidad del modelo para generalizar sobre diferentes escenarios de las imágenes satelitales.
- **Fine-Tuning:** Consideramos si realizar o no fine-tuning sobre el backbone, permitiendo ajustar sus pesos durante el entrenamiento. Esta opción resultó útil en algunos casos para mejorar la precisión en las áreas de interés.
- **Early Stopping:** Exploramos la utilización de early stopping para evitar el sobreajuste, estableciendo valores posibles de True o False.
- **Tasa de Dropout:** Definimos un rango de tasas de dropout entre 0.0 y 0.5. Esta tasa ayudó a controlar la regularización del modelo, donde una tasa alrededor de 0.3 mostró un buen balance entre la retención de información y la reducción del sobreajuste.

Entrenamiento del modelo

Compilamos el modelo U-Net utilizando el optimizador Adam con la tasa de aprendizaje seleccionada durante la optimización. Implementamos una función de pérdida combinada que integraba Binary Cross-Entropy y Dice Loss para manejar la naturaleza binaria de las máscaras y mejorar la precisión de la segmentación. Las métricas de evaluación incluyeron el coeficiente de Dice, IoU (Intersección sobre Unión), precisión, recall y F1-score, proporcionando una visión detallada del rendimiento del modelo.

Para controlar y mejorar el proceso de entrenamiento, integramos varios callbacks:

- **ReduceLROnPlateau:** Ajustaba la tasa de aprendizaje en función de la mejora de la pérdida de validación.
- **EarlyStopping:** Interrumpía el entrenamiento si la pérdida de validación no mejoraba tras un número específico de épocas.

- **CustomCheckpoint:** Implementamos un callback personalizado para guardar el modelo con el mejor desempeño en términos de varias métricas de validación (pérdida, IoU y coeficiente de Dice).
- **LastEpochCallback:** Un callback adicional que registraba la última época alcanzada, facilitando el análisis posterior del proceso de entrenamiento.

Evaluación y generación de predicciones

Tras el entrenamiento, evaluamos el modelo utilizando un conjunto de datos de prueba previamente reservado. Para este fin, cargamos el modelo entrenado y configuramos un generador de datos para las imágenes y máscaras de prueba. Posteriormente, generamos las predicciones y las comparamos con las máscaras reales, evaluando el rendimiento mediante las métricas seleccionadas.

Adicionalmente, visualizamos las predicciones junto con las imágenes originales y sus máscaras verdaderas. Esto permitió realizar un análisis cualitativo del desempeño del modelo en la segmentación de las áreas de bosques y cuerpos de agua. Las predicciones generadas se guardaron en un directorio específico para referencia y análisis futuros.

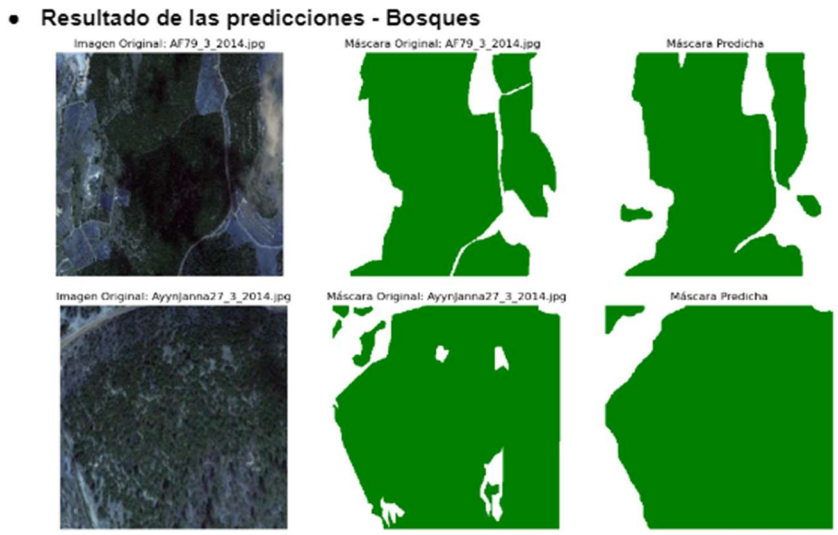

Figura 10: Ejemplo primeras segmentaciones realizadas Bosques

- **Resultado de las predicciones - Aguas**

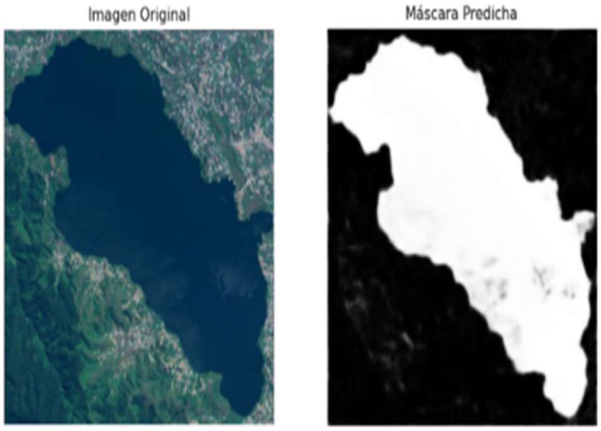

Figura 11: Ejemplo primeras segmentaciones realizadas Agua

6. RESULTADOS

Para cumplir con los objetivos de nuestra tesis, que consistió en implementar y entrenar un modelo de aprendizaje automático para identificar y clasificar áreas de bosques y cuerpos de agua a partir de imágenes satelitales, desarrollamos un enfoque meticuloso y detallado que abarcó desde la preparación de los datos hasta la generación de predicciones.

El primer paso fue la reducción de todas las imágenes satelitales a un formato estándar de 256x256 píxeles. Este proceso fue fundamental para optimizar el entrenamiento del modelo, ya que redujo considerablemente la carga computacional y el tiempo de procesamiento sin comprometer en exceso la calidad de las imágenes. Esta estrategia permitió un procesamiento más eficiente al trabajar con un gran volumen de datos.

clasificar las áreas de bosques y cuerpos de agua a partir de imágenes satelitales. Se utilizó una arquitectura U-Net adaptada para la segmentación binaria, con la capacidad de identificar áreas correspondientes a cuerpos de agua y bosques.

Segmentación de cuerpos de agua: En el caso de las imágenes de agua, el modelo estaba diseñado para distinguir entre el agua y el fondo, representado por un esquema en blanco y negro. Las máscaras utilizadas para este modelo fueron leídas en formato de escala de grises y normalizadas a valores binarios (0 y 1). El modelo segmentaba correctamente el agua como áreas en blanco (1), y el fondo como áreas en negro (0).

Segmentación de bosques: Para las imágenes de bosques, el enfoque fue más detallado. En lugar de utilizar un esquema binario de escala de grises, se empleó un método basado en la identificación de tonos de verde específicos. Las máscaras se leyeron en formato de color y se redimensionaron para asegurar uniformidad en el tamaño de entrada del modelo. Se diseñó un esquema de segmentación que detectaba el verde [44, 160, 44] como el color predominante en las áreas boscosas, diferenciando los píxeles verdes del fondo blanco. Este proceso permitió una segmentación precisa de las áreas de vegetación, considerando la diversidad de tonos de verde en las imágenes de satélites.

Comenzamos el proceso fijando una semilla para los generadores de números aleatorios en Python, NumPy y TensorFlow, garantizando así la reproducibilidad de nuestros experimentos. A continuación, procedimos a cargar las imágenes satelitales y sus respectivas máscaras desde los directorios designados, utilizando un generador de datos personalizado. Este generador se encargó de cargar las imágenes y las máscaras en lotes, facilitando la gestión de la memoria durante el entrenamiento del modelo. Para asegurar la correcta carga de los datos, realizamos una visualización de las primeras imágenes junto con sus máscaras, verificando que estuvieran correctamente alineadas.

- **Primeras imágenes de entrenamiento:**

  Bosques:

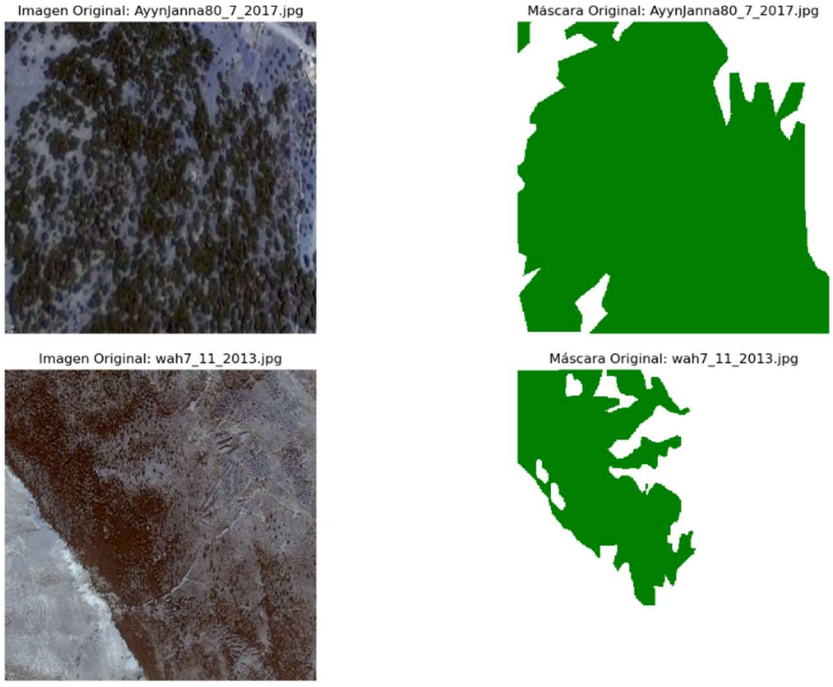

Figura 12: Ejemplo imágenes de entrenamiento Bosques

Aguas:

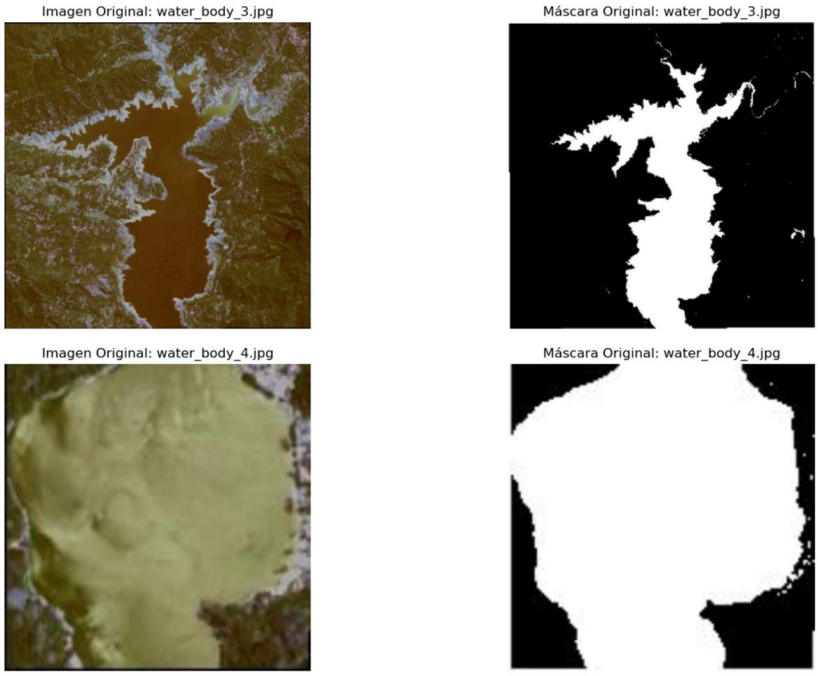

Figura 13: Ejemplo imágenes de entrenamiento Agua

- **Primeras imágenes de validación:**

Bosques:

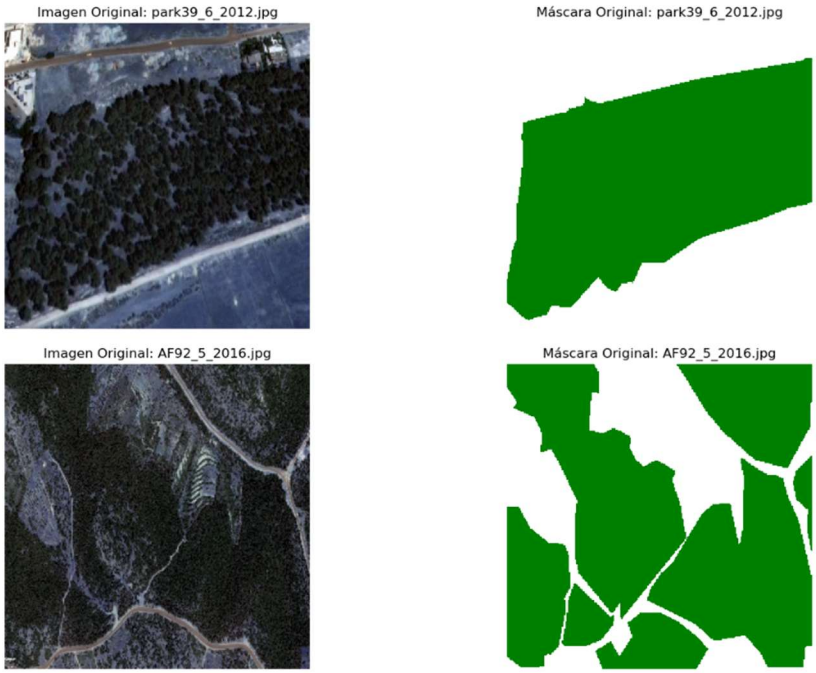

Figura 14: Ejemplo imágenes y máscaras para validación bosques

Aguas:

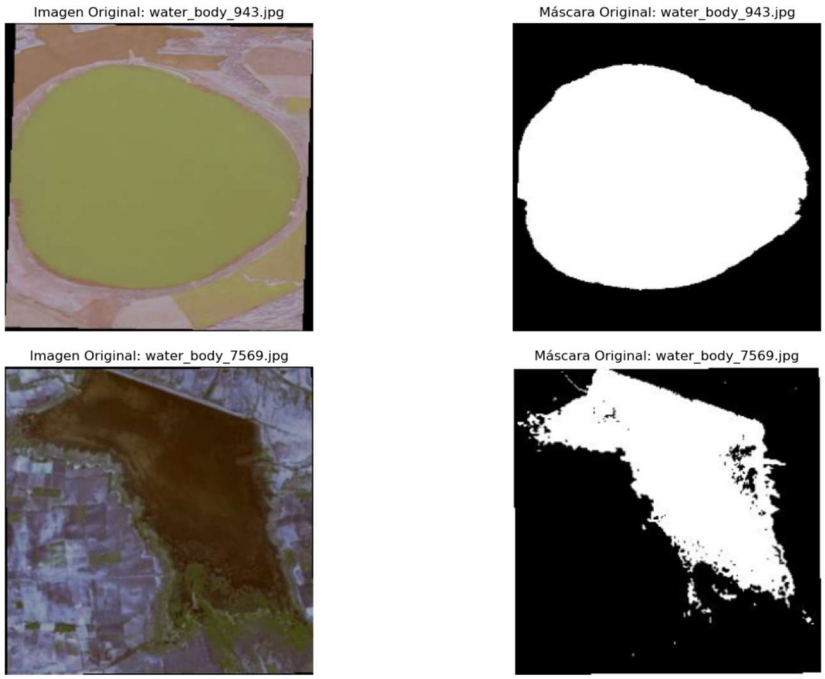

Figura 15: Ejemplo imágenes y máscaras para validación agua

En esta primera etapa del proyecto, se desarrolló y verificó el proceso de carga y visualización de las máscaras para las imágenes de cuerpos de agua y bosques. Las máscaras de agua se procesaron en escala de grises y se normalizaron a valores binarios, mientras que las máscaras de bosques se cargaron en formato RGB, detectando el tono verde [44, 160, 44] para representar la vegetación. Este proceso asegura la correcta preparación de los datos para el modelado posterior, garantizando la coherencia en las entradas y facilitando la validación visual de las áreas segmentadas antes del entrenamiento.

Para abordar el problema de clasificación, seleccionamos el modelo U-Net, conocido por su efectividad en tareas de segmentación de imágenes. Decidimos utilizar backbones preentrenados, como ResNet34, EfficientNetB3 y VGG16, con pesos iniciales de ImageNet, lo que ayudó a acelerar el entrenamiento y mejorar el rendimiento del modelo. Además, agregamos una capa de dropout en el decodificador del U-Net para reducir el riesgo de sobreajuste. La tasa de dropout se configuró como un hiperparámetro a optimizar para encontrar el equilibrio adecuado entre regularización y capacidad de generalización.

Optimización de hiperparámetros

La optimización de los hiperparámetros fue fundamental para mejorar el rendimiento del modelo. Utilizamos la herramienta **Optuna** para llevar a cabo la búsqueda de la mejor combinación de hiperparámetros a través de **múltiples ensayos** (trials). Los hiperparámetros que optimizamos incluyeron:

- **Tasa de aprendizaje:** Exploramos valores en el rango **[1e-5, 1e-3]** en escala logarítmica. Optuna seleccionó valores como **1e-4** y **5e-5**, los cuales contribuyeron a una convergencia más precisa del modelo.
- **Backbone:** Optimizamos la elección del backbone entre **ResNet34**, **EfficientNetB3** y **VGG16**, evaluando su impacto en el desempeño general del modelo.
- **Data Augmentation:** Evaluamos si aplicar o no técnicas de data augmentation, con valores posibles de **True** o **False**. La inclusión de data augmentation en algunos ensayos mejoró la capacidad del modelo para generalizar sobre diferentes escenarios de las imágenes satelitales.
- **Fine-Tuning:** Consideramos si realizar o no fine-tuning sobre el backbone, permitiendo **ajustar sus pesos durante el entrenamiento**. Esta opción resultó útil en algunos casos para mejorar la precisión en las áreas de interés.
- **Early Stopping:** Exploramos la utilización de early stopping para **evitar el sobreajuste**, estableciendo valores posibles de **True** o **False**.
- **Tasa de Dropout:** Definimos un rango de tasas de dropout entre **0.0** y **0.5**. Esta tasa ayudó a controlar la regularización del modelo, donde una tasa alrededor de **0.3** mostró un buen balance entre la retención de información y la reducción del sobreajuste.

Entrenamiento del modelo

Compilamos el modelo U-Net utilizando el optimizador **Adam** con la tasa de aprendizaje seleccionada durante la optimización. Implementamos una función de pérdida combinada que integraba **Binary Cross-Entropy** y **Dice Loss** para manejar la naturaleza binaria de las máscaras y mejorar la precisión de la segmentación. Las métricas de evaluación incluyeron **el coeficiente de Dice**, **IoU** (Intersección sobre Unión), **precisión**, **recall** y **F1-score**, proporcionando una visión detallada del rendimiento del modelo.

Para **controlar y mejorar el proceso de entrenamiento**, integramos varios **callbacks**:

- **ReduceLROnPlateau:** Ajustaba la tasa de aprendizaje en función de la mejora de la pérdida de validación.
- **EarlyStopping:** Interrumpía el entrenamiento si la pérdida de validación no mejoraba tras un número específico de épocas.

- **CustomCheckpoint:** Implementamos un **callback personalizado** para guardar el modelo con el mejor desempeño en términos de varias **métricas de validación** (**pérdida**, **IoU** y **coeficiente de Dice**).
- **LastEpochCallback:** Un callback adicional que registraba la última época alcanzada, facilitando el análisis posterior del proceso de entrenamiento.

Evaluación y generación de predicciones

Tras el entrenamiento, evaluamos el modelo utilizando un conjunto de datos de prueba previamente reservado. Para este fin, cargamos el modelo entrenado y configuramos un generador de datos para las imágenes y máscaras de prueba. Posteriormente, generamos las predicciones y las comparamos con las máscaras reales, evaluando el rendimiento mediante las métricas seleccionadas.

Adicionalmente, visualizamos las predicciones junto con las imágenes originales y sus máscaras verdaderas. Esto permitió realizar un análisis cualitativo del desempeño del modelo en la segmentación de las áreas de bosques y cuerpos de agua. Las predicciones generadas se guardaron en un directorio específico para referencia y análisis futuros.

1. Resultado del modelo de Aguas:

- **Backbone y arquitectura**

En este caso, observamos que el backbone **EfficientNetB3** ha demostrado ser el más efectivo para la tarea de segmentación de cuerpos de agua en comparación con otros backbones, como **ResNet34** y **VGG16**. Esto tiene sentido desde una perspectiva de diseño, ya que **EfficientNet** utiliza una estrategia de scaling óptima (depth, width, resolution), lo que le permite manejar tanto detalles finos como características globales en las imágenes, cruciales para la segmentación precisa de cuerpos de agua.

El hecho de que **EfficientNetB3** haya superado a otros backbones puede estar relacionado con su capacidad de extraer mejor las características clave, permitiendo una segmentación más precisa en límites complejos de agua.

- **Métricas clave: IoU y F1-Score**

El **IoU** (Intersection over Union) es una métrica crítica para segmentación, ya que mide cuánto se solapan las predicciones con las áreas verdaderas en las máscaras. Un **IoU** de **0.8016** en el modelo más destacado (**ensayo 25**) indica un excelente nivel de precisión en la segmentación, donde más del **80%** del área predicha por el modelo coincide con las áreas de agua reales en las imágenes.

Por otro lado, el **F1-score** de **0.9048** demuestra que el modelo logra un buen equilibrio entre la

precisión (**0.9307**) y el **recall** (**0.8836**). La precisión alta indica que el modelo comete pocos falsos positivos (predice agua donde no hay), mientras que el **recall** más moderado sugiere que el modelo podría omitir algunas áreas más pequeñas o difíciles de detectar. Sin embargo, este balance es adecuado en escenarios donde se prioriza la precisión para minimizar las áreas incorrectamente clasificadas como agua.

- **Configuración del modelo**

Un aspecto notable es que el modelo con mejores resultados (ensayo 25) **no utilizó data augmentation ni fine-tuning** del backbone. Esta observación es interesante porque, en muchos casos, la **data augmentation** es clave para mejorar la generalización de los modelos de segmentación, especialmente en tareas con variabilidad en las imágenes, como el análisis satelital. La ausencia de **data augmentation** podría significar que las características de las imágenes de cuerpos de agua en este **conjunto de datos son relativamente consistentes**, y el **modelo fue capaz de aprender** las representaciones necesarias sin necesidad de variaciones adicionales.

Además, el **learning rate** inicial de **0.000575** y el ajuste final de **2.3e-5** indican que el modelo fue entrenado de manera cuidadosa, ajustando el ritmo de aprendizaje a medida que avanzaba el entrenamiento, lo que le permitió alcanzar la convergencia sin sobreajustarse.

- **Comparación con otros ensayos**

Aunque varios ensayos con **EfficientNetB3** obtuvieron resultados similares, como los ensayos **18**, **17**, y **23**, el ensayo **25 logró una combinación óptima de métricas en solo 22 épocas**, lo cual es un indicador positivo de eficiencia. La estabilidad en las métricas clave a lo largo de estos ensayos refuerza la idea de que **EfficientNetB3** es un backbone particularmente bien adaptado para la segmentación de cuerpos de agua, probablemente debido a su capacidad de manejar variaciones de escala y detalles complejos en las fronteras de las áreas de agua.

- **Implicaciones Prácticas**

Desde una perspectiva práctica, el **modelo del ensayo 25** no solo proporciona una segmentación precisa, sino que también lo hace de manera eficiente, requiriendo menos épocas para alcanzar buenos resultados. Este tipo de eficiencia es crucial en escenarios del mundo real, donde los recursos computacionales y el tiempo de entrenamiento pueden ser limitados. Además, la configuración sin data augmentation podría indicar que el modelo está bien adaptado a este conjunto de datos específico, aunque se debería evaluar su capacidad de generalización en nuevos datos o entornos más variados.

- **Métricas del mejor modelo y predicciones**

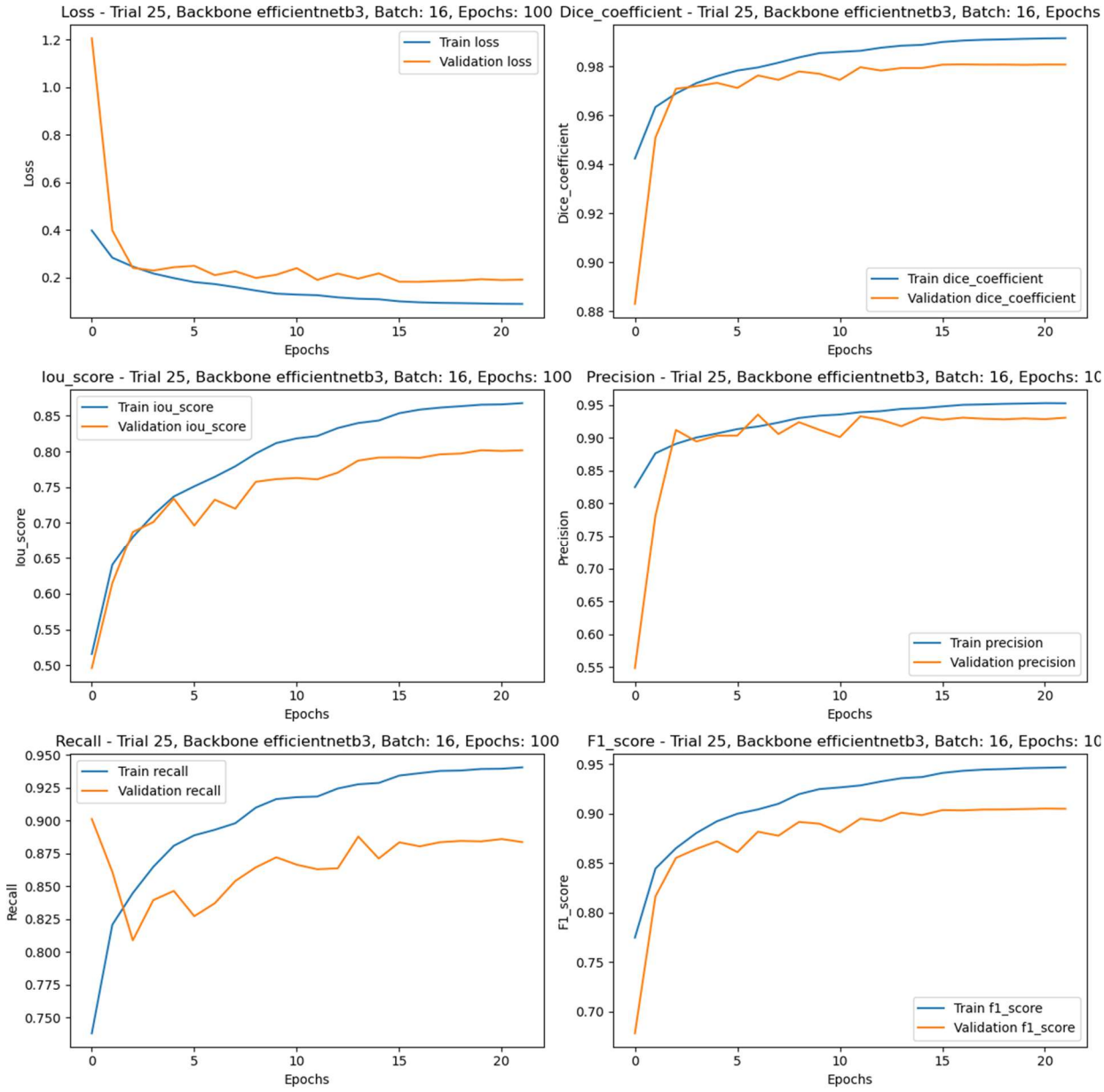

Figura 16: Métricas modelo Efficientnetb3 Agua

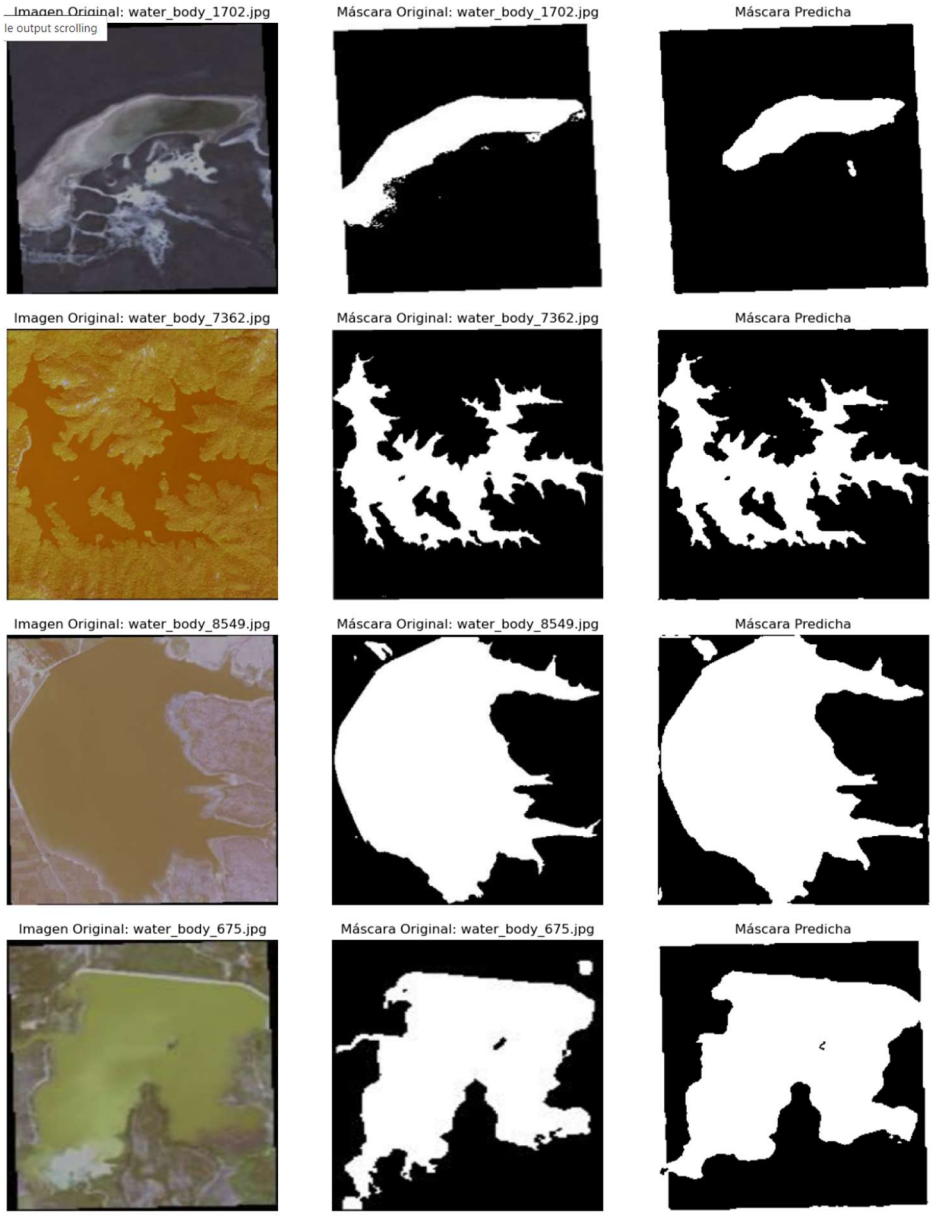

Figura 17: Ejemplos de segmentación agua, modelo efficientnetb3

En conclusión, el modelo basado en EfficientNetB3 en el ensayo 25 es el mejor entre los evaluados, ya que ofrece un equilibrio óptimo entre precisión y recall, con un excelente IoU de 0.8016 y un F1-score de 0.9048. Este rendimiento se logró sin la necesidad de data augmentation o fine-tuning, lo que sugiere que el backbone EfficientNetB3 está bien adaptado para esta tarea de segmentación. La eficiencia del modelo, alcanzando estos resultados en solo 22 épocas, lo convierte en la mejor opción para aplicaciones prácticas de segmentación de cuerpos de agua.

2. Resultado del modelo de Bosques:

- **Backbone y Arquitectura**

Al analizar los resultados, es evidente que el modelo con **EfficientNetB3** en el **ensayo número 5** obtuvo el **mejor rendimiento** general. **EfficientNetB3** se destacó frente a otros backbones como **ResNet34** y **VGG16**, lo cual es consistente con su capacidad de equilibrar el detalle y la resolución en las características de las imágenes, algo clave para la segmentación precisa de áreas de vegetación. En el contexto de bosques, donde la variabilidad de los tonos de verde es alta y las fronteras entre vegetación y otros elementos pueden ser difusas, la capacidad de **EfficientNetB3** para extraer características multiescalares es crucial.

- **Métricas Clave: IoU y F1-Score**

El **IoU** (Intersection over Union) para el **modelo del ensayo 5** es de **0.9502**, lo que indica una sobresaliente capacidad para capturar las áreas de bosque en las imágenes. Este valor de **IoU** es significativamente alto, lo que implica que el modelo tiene un excelente nivel de solapamiento entre las predicciones y las áreas reales de bosque. El **F1-score** de **0.9834** también es muy alto, lo que refleja un buen equilibrio entre **precisión** (**0.9815**) y **recall** (**0.9852**). Esto significa que el modelo es capaz de identificar correctamente la gran mayoría de las áreas de vegetación, minimizando los falsos negativos y logrando una alta sensibilidad en la detección de los bosques.

- **Configuración del Modelo**

Al igual que en el análisis de cuerpos de agua, resulta notable que el modelo con mejores resultados (**ensayo 5**) **no utilizó data augmentation ni fine-tuning** del backbone. Este hecho es particularmente interesante dado que las imágenes de bosques, por su naturaleza, presentan más variabilidad en los tonos de verde y las estructuras. **La ausencia de data augmentation** podría indicar que las imágenes utilizadas en este conjunto de datos tienen características consistentes, permitiendo al modelo aprender de manera eficaz sin la necesidad de transformaciones adicionales. Sin embargo, en escenarios con mayor diversidad, la inclusión de data augmentation podría ser beneficiosa.

En cuanto al **learning rate inicial** de **0.000698**, y el **ajuste final** a **1.39e-04** en la **época 12**, este modelo logró una rápida convergencia, lo que sugiere que el ajuste del ritmo de aprendizaje fue apropiado para el conjunto de datos y permitió que el modelo alcanzara un excelente rendimiento en un **tiempo relativamente corto**.

- **Comparación con otros ensayos**

Si bien otros modelos, como el **ensayo 1** con **ResNet34**, lograron un **IoU** cercano a **0.9467** y un **F1-**

score de **0.9840**, **EfficientNetB3** del **ensayo 5** sigue destacándose ligeramente en términos de **convergencia rápida** y **eficiencia**. La capacidad de **EfficientNetB3** para alcanzar altos valores de **IoU** y **F1-score** en **menos épocas** (12 en comparación con 100) refuerza su eficiencia como modelo.

Además, el hecho de que otros ensayos con **VGG16** también obtuvieron resultados competitivos muestra que **VGG16** es robusto para la tarea, aunque **EfficientNetB3** mantiene una **ligera ventaja** en términos de rendimiento general.

- **Implicaciones Prácticas**

En términos prácticos, el modelo basado en **EfficientNetB3** no solo ofrece una alta **precisión** y **recall**, sino que lo hace en **menos épocas**, lo que sugiere una mayor eficiencia en términos de tiempo de entrenamiento. Esta eficiencia es crucial en aplicaciones donde los recursos computacionales y el tiempo son limitados. Además, el modelo **sin data augmentation ni fine-tuning** parece estar bien adaptado al conjunto de datos específico, lo que podría simplificar su implementación en entornos controlados.

No obstante, en escenarios donde se espere mayor variabilidad en las imágenes o condiciones más desafiantes (como iluminación variable o cambios estacionales en los bosques), podría ser útil explorar el uso de data augmentation para mejorar la generalización.

- **Métricas del mejor modelo y predicciones**

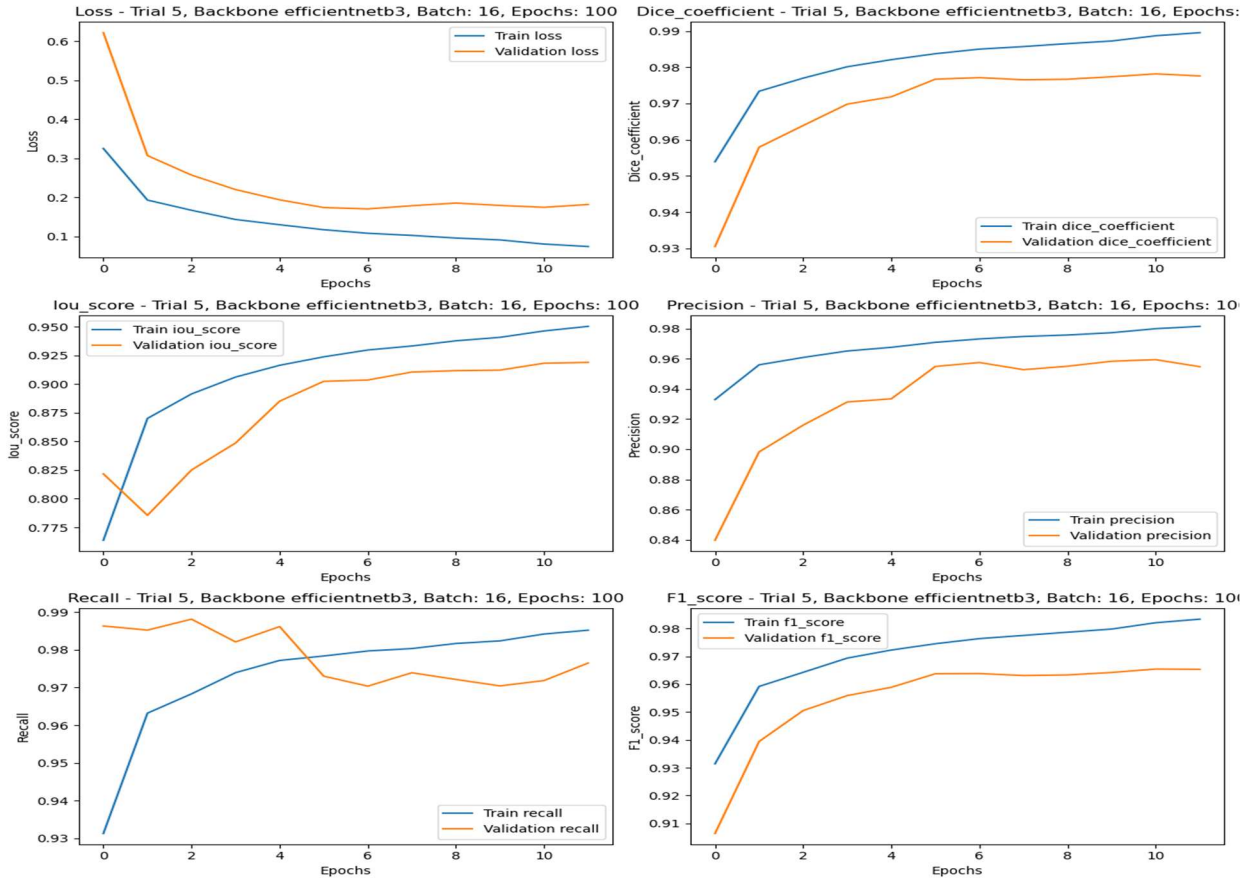

Figura 18: Métricas modelo efficientnetb3 Bosques

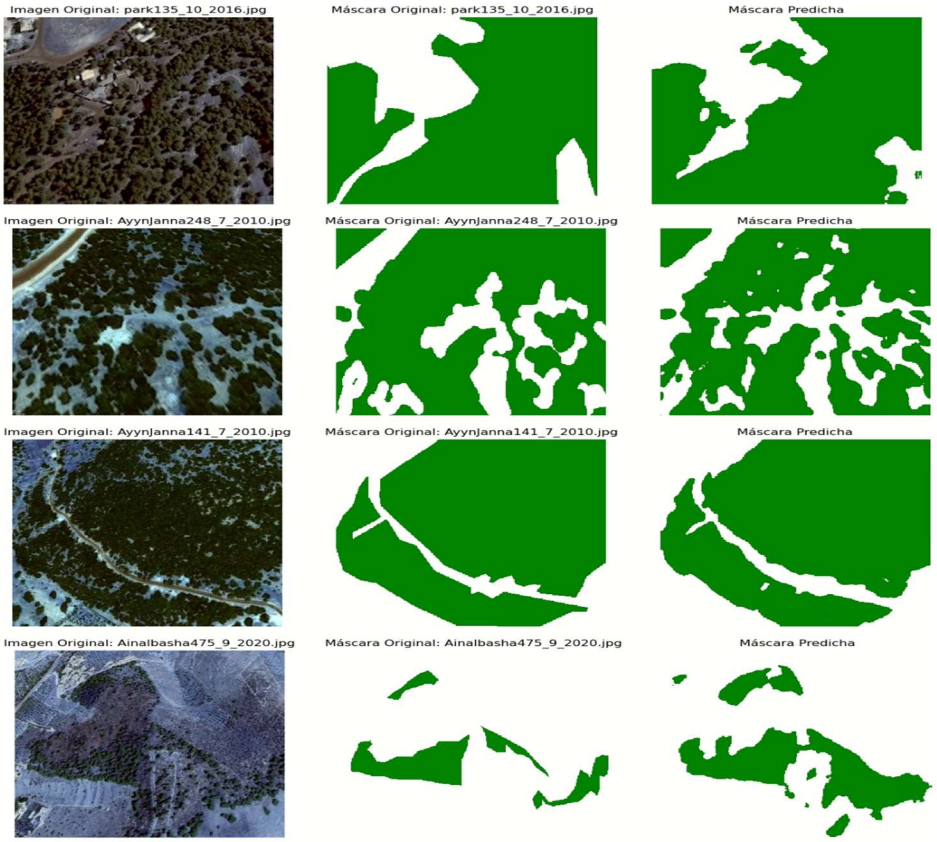

Figura 19: Ejemplos segmentación Bosques, efficientnetb3

En conclusión, el modelo con EfficientNetB3 del ensayo 5 es el más eficiente y preciso en términos de segmentación de áreas de bosque, con un excelente IoU de 0.9467 y un F1-score de 0.9840. Este modelo fue capaz de lograr estos resultados en solo 12 épocas, lo que lo convierte en una opción altamente eficiente.

TABLAS DE RESULTADOS

BOSQUES

trial_number	backbone	Architecture	Batch Size	Epochs	learning_rate_inicial	Early Stopping	Fine-tuned backbone	Data Augmentation	Clase	iou_score	f1_score
0	resnet34	UNet	16	100	5.61152E-05	no	no	si	Bosques	0.892061532	0.991231501
1	resnet34	UNet	16	100	0.000870602	no	si	no	Bosques	0.946786284	0.98237443
2	vgg16	UNet	16	100	3.83963E-05	si	si	no	Bosques	0.942642331	0.988919973
3	resnet34	UNet	16	100	0.000790262	si	si	no	Bosques	0.932364464	0.97829932
4	resnet34	UNet	16	100	3.29276E-05	si	si	si	Bosques	0.815183222	0.972636819
5	**efficientnetb3**	**UNet**	**16**	**100**	**0.000697828**	**si**	**no**	**no**	**Bosques**	**0.95023787**	**0.983361542**
6	vgg16	UNet	16	100	1.91359E-05	no	no	si	Bosques	0.932789862	0.994223237
7	vgg16	UNet	16	100	1.40634E-05	no	no	si	Bosques	0.919609487	0.994699717
8	vgg16	UNet	16	100	0.000594875	si	si	si	Bosques	0.926867962	0.97604239
9	efficientnetb3	UNet	16	100	1.64355E-05	si	no	no	Bosques	0.909871757	0.986813605
10	vgg16	UNet	16	100	0.000217463	si	si	si	Bosques	0.925453842	0.978224277
11	vgg16	UNet	16	100	0.000220443	si	si	si	Bosques	0.926069021	0.978380263
12	vgg16	UNet	16	100	0.000269354	si	si	si	Bosques	0.907319188	0.97022754
13	vgg16	UNet	16	100	0.000254612	si	si	si	Bosques	0.926603556	0.976002693
14	vgg16	UNet	16	100	0.000120016	si	si	si	Bosques	0.931398451	0.981858313
15	vgg16	UNet	16	100	0.000416804	si	si	si	Bosques	0.922976971	0.976162195
16	efficientnetb3	UNet	16	100	0.000123623	si	si	si	Bosques	0.947873473	0.987815917
17	vgg16	UNet	16	100	0.000423636	si	si	si	Bosques	0.927849889	0.976886332
18	vgg16	UNet	16	100	0.000379006	no	si	si	Bosques	0.948091805	0.983076036
19	efficientnetb3	UNet	16	100	0.000166879	si	si	si	Bosques	0.952946723	0.988492489
20	vgg16	UNet	16	100	7.26195E-05	si	si	si	Bosques	0.91817385	0.986525595
21	vgg16	UNet	16	100	0.000426959	si	si	si	Bosques	0.916176856	0.974176943
22	vgg16	UNet	16	100	0.000423962	si	si	si	Bosques	0.922085702	0.97455734
23	vgg16	UNet	16	100	0.000180533	si	si	si	Bosques	0.937043905	0.980521321
24	vgg16	UNet	16	100	0.000332234	si	si	si	Bosques	0.932260633	0.977700591
25	vgg16	UNet	16	100	0.000574588	si	si	si	Bosques	0.914861858	0.971436977
26	vgg16	UNet	16	100	0.000155822	no	no	no	Bosques	0.943653643	0.983294964
27	vgg16	UNet	16	100	0.000480782	si	si	si	Bosques	0.93970716	0.979831398
28	resnet34	UNet	16	100	0.000294819	si	si	si	Bosques	0.94661504	0.984046221
29	efficientnetb3	UNet	16	100	7.70659E-05	no	no	si	Bosques	0.948538601	0.992064297

Tabla 3: Resultados modelos para segmentación de Agua

AGUAS

trial_number	backbone	Architecture	Batch Size	Epochs	learning_rate_inicial	Early Stopping	Fine-tuned backbone	Data Augmentation	Clase	iou_score	f1_score
0	resnet34	UNet	16	100	5.61152E-05	no	no	si	Agua	0.692388535	0.834393144
1	resnet34	UNet	16	100	0.000870602	no	si	no	Agua	0.751447618	0.880724907
2	vgg16	UNet	16	100	3.83963E-05	si	si	no	Agua	0.732548773	0.873900354
3	resnet34	UNet	16	100	0.000790262	si	si	no	Agua	0.733451188	0.874964893
4	resnet34	UNet	16	100	3.29276E-05	si	si	si	Agua	0.576478124	0.784685612
5	efficientnetb3	UNet	16	100	0.000697828	si	no	no	Agua	0.79073751	0.90119338
6	vgg16	UNet	16	100	1.91359E-05	no	no	si	Agua	0.712728739	0.865095496
7	vgg16	UNet	16	100	1.40634E-05	no	no	si	Agua	0.711002946	0.862682402
8	vgg16	UNet	16	100	0.000594875	si	si	si	Agua	0.652249575	0.824732542
9	efficientnetb3	UNet	16	100	1.64355E-05	si	no	no	Agua	0.707211912	0.86710304
10	efficientnetb3	UNet	16	100	0.000211413	si	no	no	Agua	0.775086761	0.893068671
11	efficientnetb3	UNet	16	100	0.000226735	si	no	no	Agua	0.777627647	0.894291878
12	efficientnetb3	UNet	16	100	0.000199294	si	no	no	Agua	0.773273408	0.893961072
13	efficientnetb3	UNet	16	100	0.000279846	si	no	no	Agua	0.779843032	0.894396245
14	efficientnetb3	UNet	16	100	0.000120016	si	no	no	Agua	0.773817062	0.892237246
15	efficientnetb3	UNet	16	100	0.000416804	si	no	no	Agua	0.793223441	0.90196389
16	efficientnetb3	UNet	16	100	0.000377091	si	no	no	Agua	0.787220657	0.899421811
17	efficientnetb3	UNet	16	100	0.00045542	si	no	no	Agua	0.795404673	0.901949286
18	efficientnetb3	UNet	16	100	0.00040674	no	no	no	Agua	0.798709095	0.901701868
19	efficientnetb3	UNet	16	100	0.000129383	si	no	no	Agua	0.775488496	0.891596794
20	efficientnetb3	UNet	16	100	0.00095138	si	no	no	Agua	0.78396821	0.900332391
21	efficientnetb3	UNet	16	100	0.000910354	si	no	no	Agua	0.789802849	0.903073907
22	efficientnetb3	UNet	16	100	0.000996675	si	no	no	Agua	0.789779067	0.900983155
23	efficientnetb3	UNet	16	100	0.000947972	si	no	no	Agua	0.794909954	0.902255356
24	efficientnetb3	UNet	16	100	0.000596288	si	no	no	Agua	0.786456943	0.899278224
25	**efficientnetb3**	**UNet**	**16**	**100**	**0.000575117**	**si**	**no**	**no**	**Agua**	**0.801582813**	**0.904791772**
26	efficientnetb3	UNet	16	100	0.000539711	no	si	si	Agua	0.786177695	0.898832798
27	efficientnetb3	UNet	16	100	0.000312311	si	no	no	Agua	0.777778983	0.897314668
28	resnet34	UNet	16	100	7.82614E-05	si	no	no	Agua	0.660980225	0.826324105
29	vgg16	UNet	16	100	0.000694327	no	no	si	Agua	0.648177862	0.815780044

Tabla 4: Resultados modelos para segmentación de Agua

7. DIAGRAMAS DE LOS MODELOS

En esta sección, se explican los principales diagramas que describen la arquitectura, flujo de datos, optimización y evaluación del modelo U-Net utilizado para la segmentación de imágenes (cuerpos de bosques y agua). Cada diagrama juega un papel clave en la comprensión de cómo se desarrolló y entrenó el modelo, así como se evaluaron los resultados.

Diagrama de arquitectura del modelo U-Net

El primer diagrama muestra la arquitectura del modelo U-Net utilizado para la segmentación de imágenes. Este modelo está compuesto por un **backbone** preentrenado (**ResNet34, EfficientNetB3 o VGG16**), que es utilizado para extraer características de las imágenes de entrada. Luego, las características extraídas pasan por el decodificador de U-Net, donde se utilizan capas de convolución, upsampling, concatenación, y Dropout para evitar el overfitting. Finalmente, la salida del modelo es una máscara de segmentación de la misma resolución que la imagen de entrada, en este caso 256x256 píxeles.

A continuación, se muestra el diagrama que ilustra esta arquitectura:

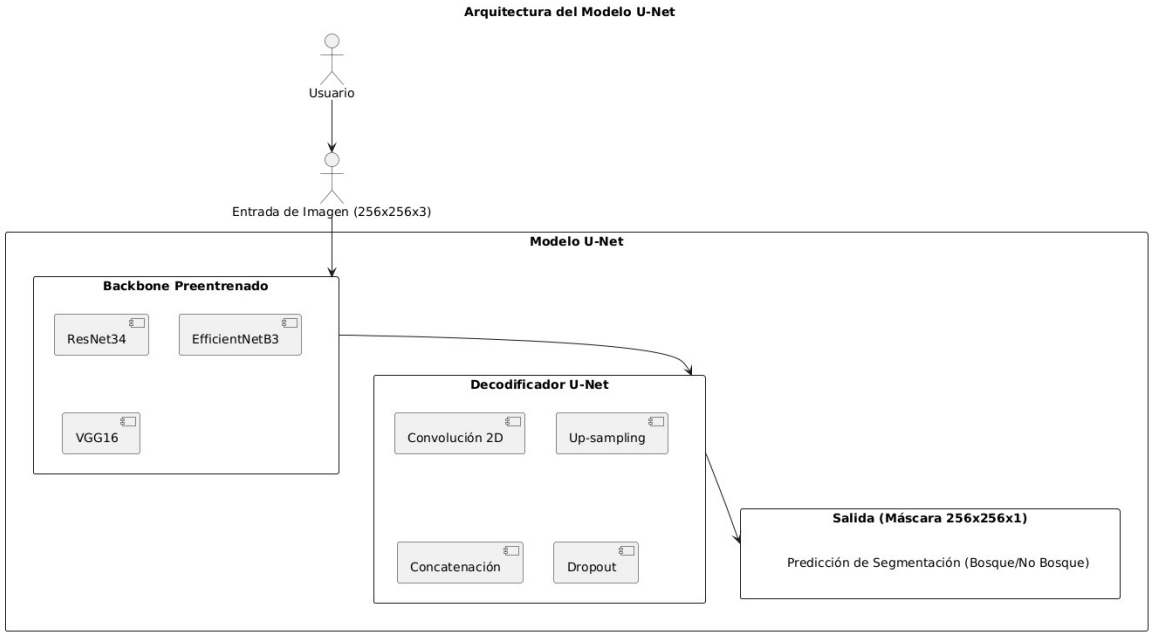

Figura 20: Flujograma, arquitectura U-net

2. Diagrama de flujo de Datos

El segundo diagrama describe el flujo de datos desde la carga de imágenes hasta la evaluación de predicciones. En este flujo, se ve cómo seleccionamos los datos de entrenamiento y validación, y cómo esos datos son procesados por el generador para crear lotes de imágenes y máscaras. El modelo U-Net es entrenado con estos datos mediante un ciclo de optimización que ajusta sus pesos. Posteriormente, las predicciones generadas por el modelo se comparan con las máscaras originales para evaluar el rendimiento.

El siguiente diagrama representa este flujo de datos:

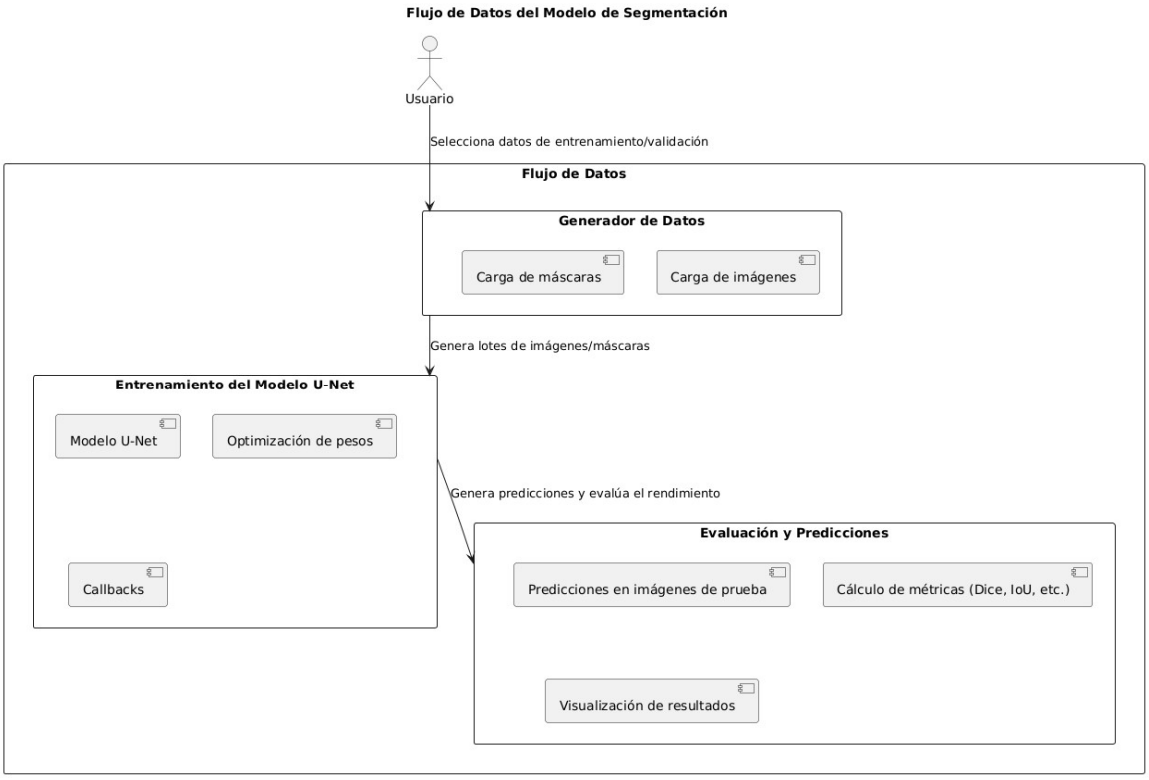

Figura 21: Flujograma, flojo de datos

3. Diagrama de optimización de hiperparámetros con Optuna

Este diagrama ilustra cómo se realizó la optimización de los hiperparámetros utilizando Optuna. Se realizan múltiples ensayos, cada uno con diferentes combinaciones de hiperparámetros como la tasa de aprendizaje (learning rate), el backbone, y la tasa de Dropout. Cada ensayo genera un modelo que se evalúa en base a las métricas de validación. El mejor modelo se selecciona automáticamente en función de estas métricas, asegurando que se optimicen los parámetros del modelo para un rendimiento óptimo.

A continuación, se muestra el diagrama de la optimización de hiperparámetros:

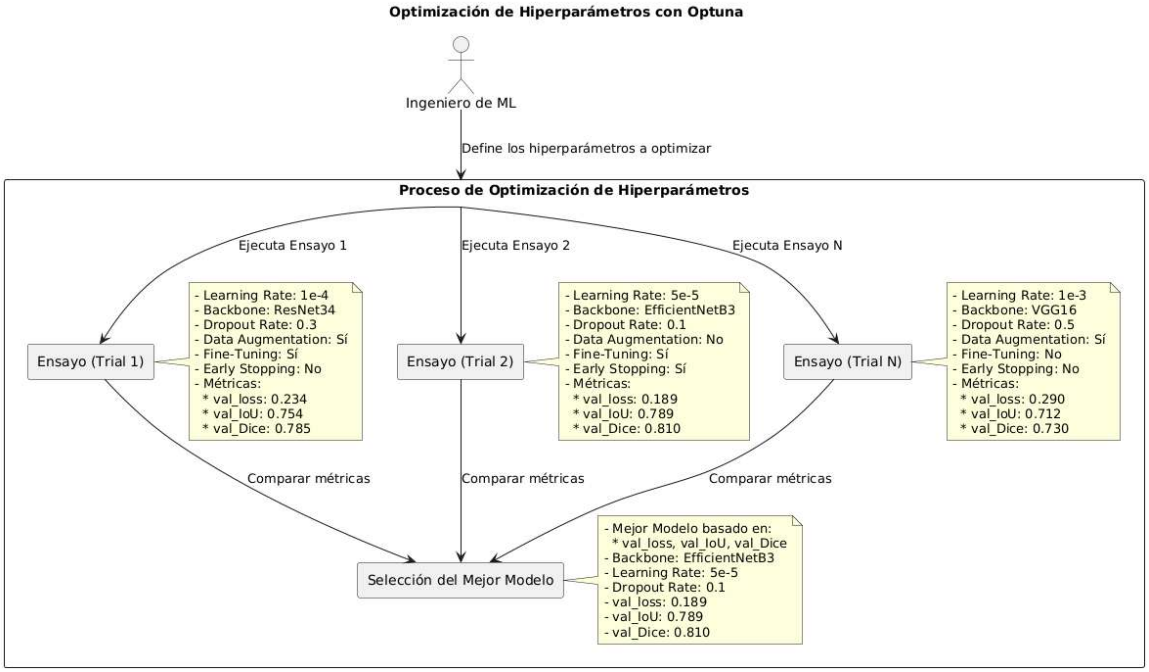

Figura 22: Flujograma optimización de parámetros.

4. Diagrama de evaluación del modelo

Este diagrama muestra cómo se evaluó el modelo U-Net entrenado utilizando un conjunto de imágenes de prueba. Las imágenes y máscaras de prueba se utilizan para generar predicciones con el modelo entrenado. Luego, estas predicciones se comparan con las máscaras originales utilizando métricas como el coeficiente de Dice, IoU, precisión, recall, y F1-Score. Finalmente, los resultados se visualizan para observar cómo se comporta el modelo en términos de segmentación de imágenes de prueba.

El diagrama de la evaluación del modelo se expone a continuación:

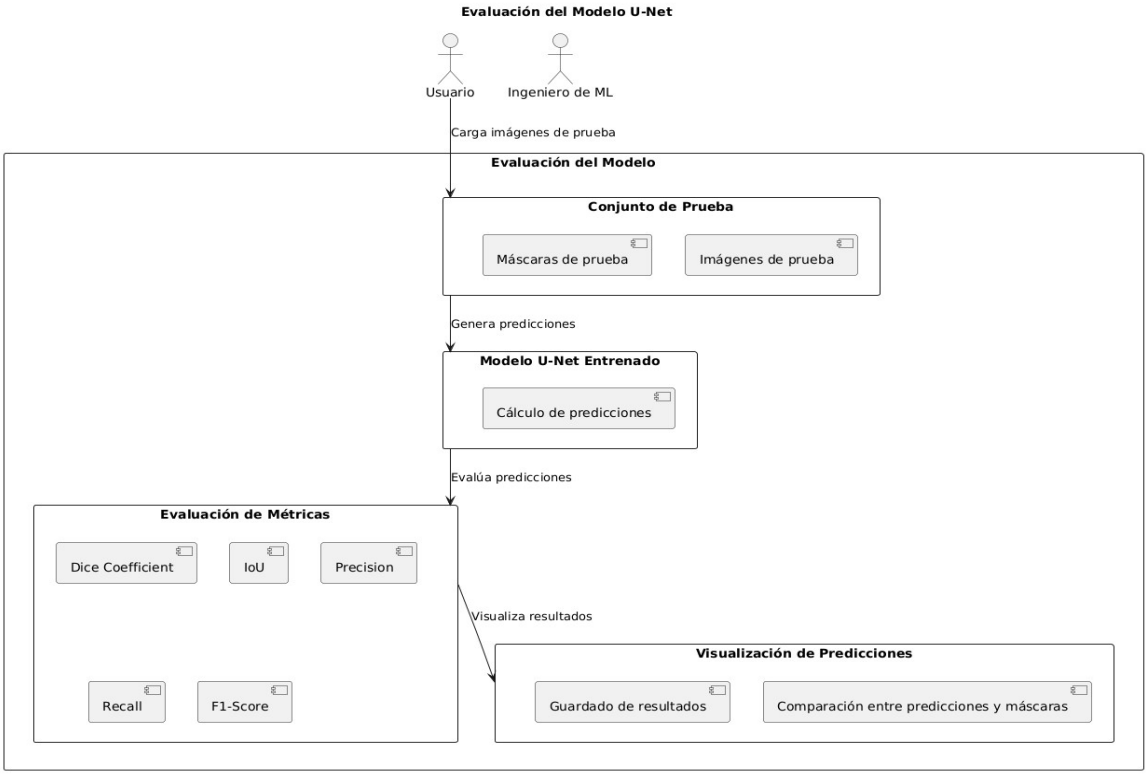

Figura 23: Flujograma evaluación de modelo U-net

5. Diagrama de funcionalidad de Callbacks

Este último diagrama explica la funcionalidad de los callbacks utilizados durante el entrenamiento del modelo U-Net. Los callbacks juegan un papel crucial en el control del proceso de entrenamiento. Por ejemplo, 'ReduceLROnPlateau' reduce la tasa de aprendizaje si no se observa una mejora en las métricas, y 'EarlyStopping' detiene el entrenamiento si no se detecta ninguna mejora después de un número determinado de épocas. Además, 'CustomCheckpoint' guarda el mejor modelo basado en las métricas de validación (como val_loss y IoU), asegurando que el mejor modelo posible se almacene para su uso posterior.

A continuación, se muestra el diagrama de la funcionalidad de los callbacks:

Figura 24: Flujograma de callbacks

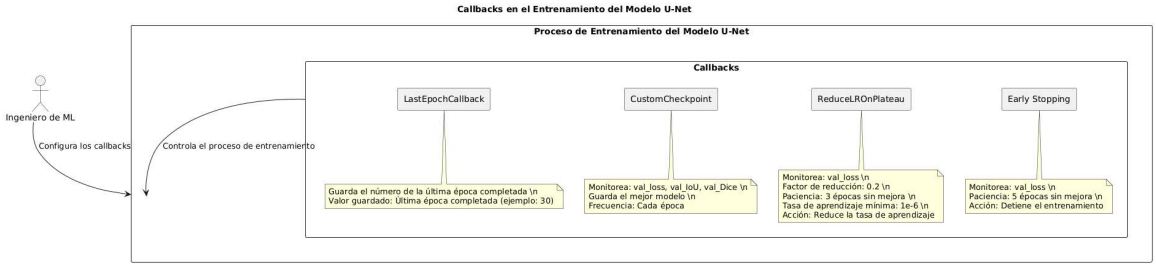

Diagrama de Clases

El diagrama de clases fue elaborado para mostrar la estructura y relaciones entre los diferentes módulos del sistema. Aunque el código estaba basado en funciones, nos permitió representar los componentes principales como clases y visualizar cómo interactuaban. Esto fue clave para identificar los módulos responsables de la carga de imágenes, predicción y procesamiento de las máscaras.

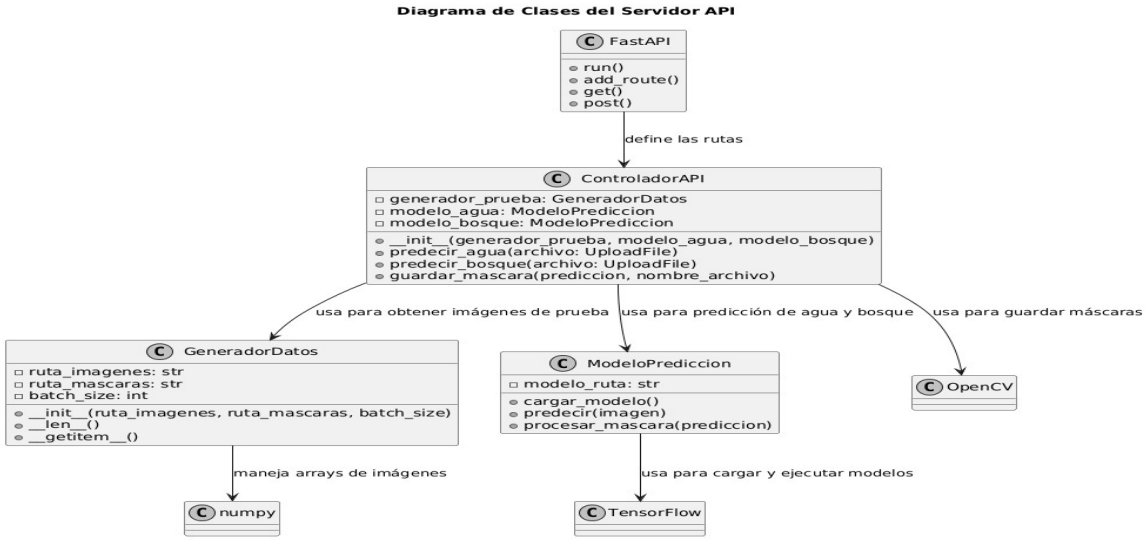

Figura 25: Flujograma funcionamiento API

Diagrama de Flujo

El diagrama de flujo nos permitió ilustrar el proceso completo desde la recepción de imágenes hasta la devolución de máscaras. A través de este diagrama, determinamos los principales puntos de decisión (validación de formato y predicción exitosa) y las diferentes etapas por las que pasaba la imagen dentro del sistema.

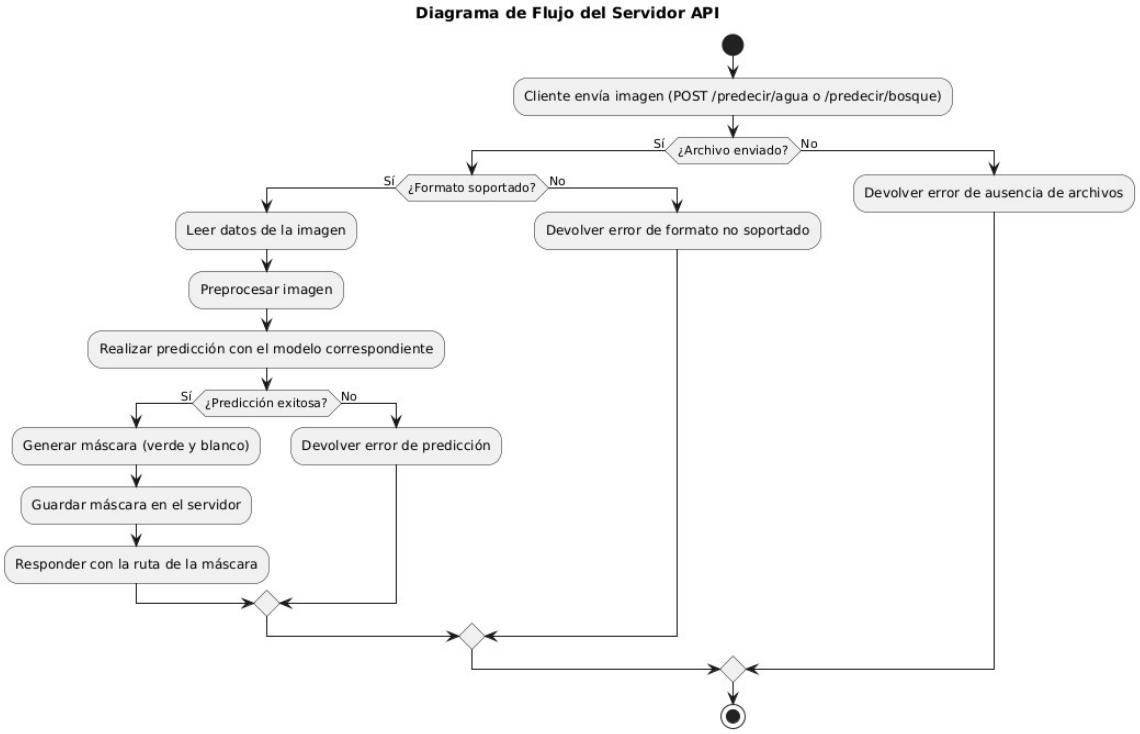

Figura 26_Flujograma procesamiento imagen desde la API

Diagrama de Secuencia

El diagrama de secuencia nos describe la interacción entre los diferentes componentes, como FastAPI, el modelo de predicción y el sistema de archivos, cuando el cliente enviaba una imagen. Este diagrama fue útil para entender el flujo de datos y cómo se transmitían las respuestas a lo largo del proceso.

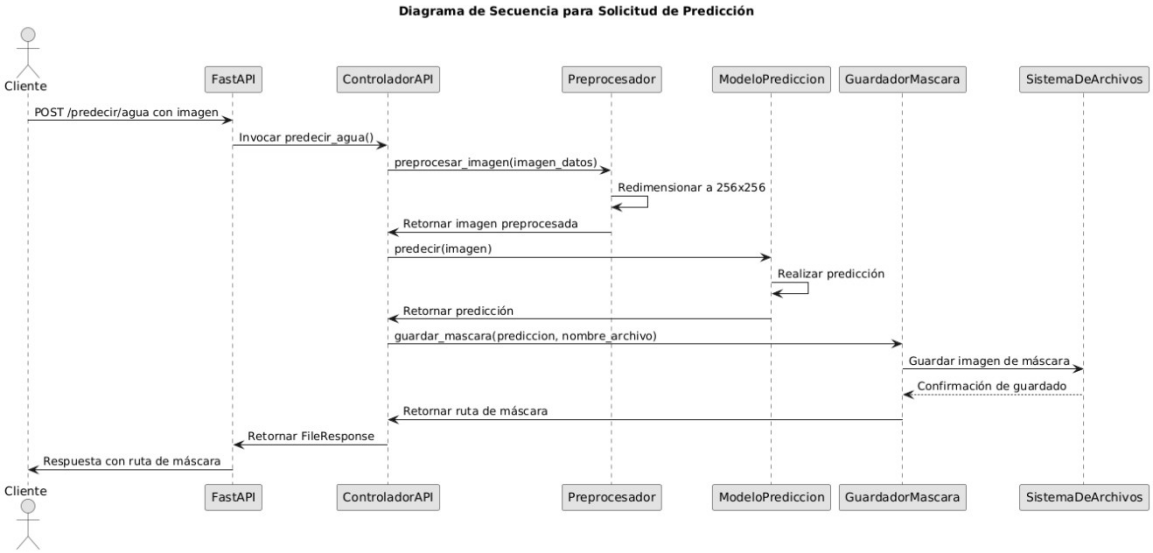

Figura 27: Flujograma generación de predicciones segmentaciones

Diagrama de componentes

El diagrama de componentes nos ayuda a desglosar la aplicación en módulos clave como FastAPI, el preprocesador de imágenes y los modelos de predicción, mostrando las dependencias entre ellos. Este diagrama fue crucial para organizar el sistema y entender cómo los módulos interactuaban entre sí para completar las solicitudes.

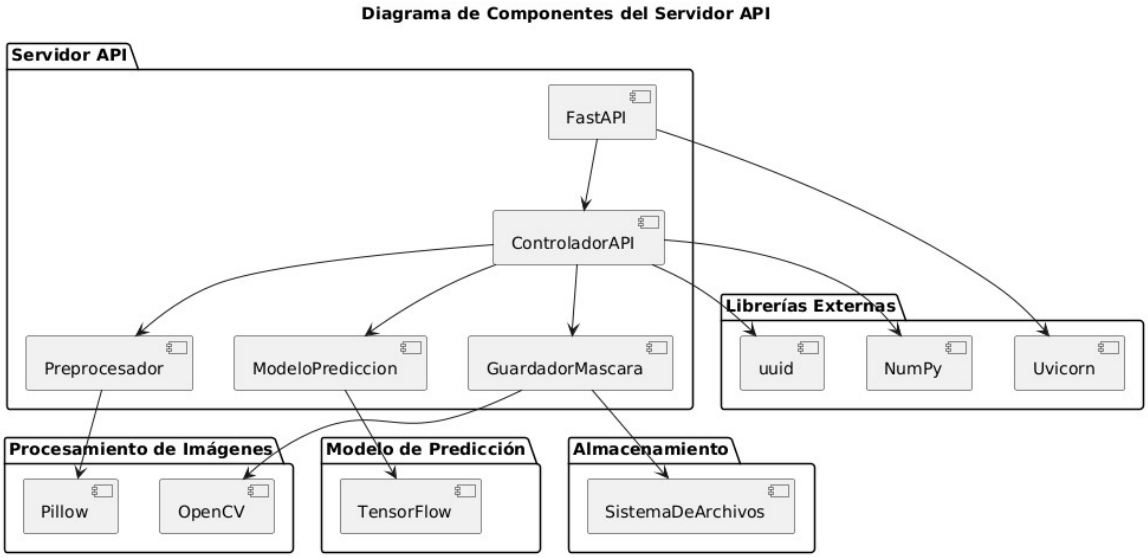

Figura 28: Flujograma componentes API

Diagrama de Despliegue

El diagrama de despliegue nos representa la arquitectura del sistema, mostrando cómo los componentes como **FastAPI** y los modelos de **TensorFlow** son desplegados en el servidor. Esto nos permitió detallar la infraestructura necesaria para correr el sistema en producción.

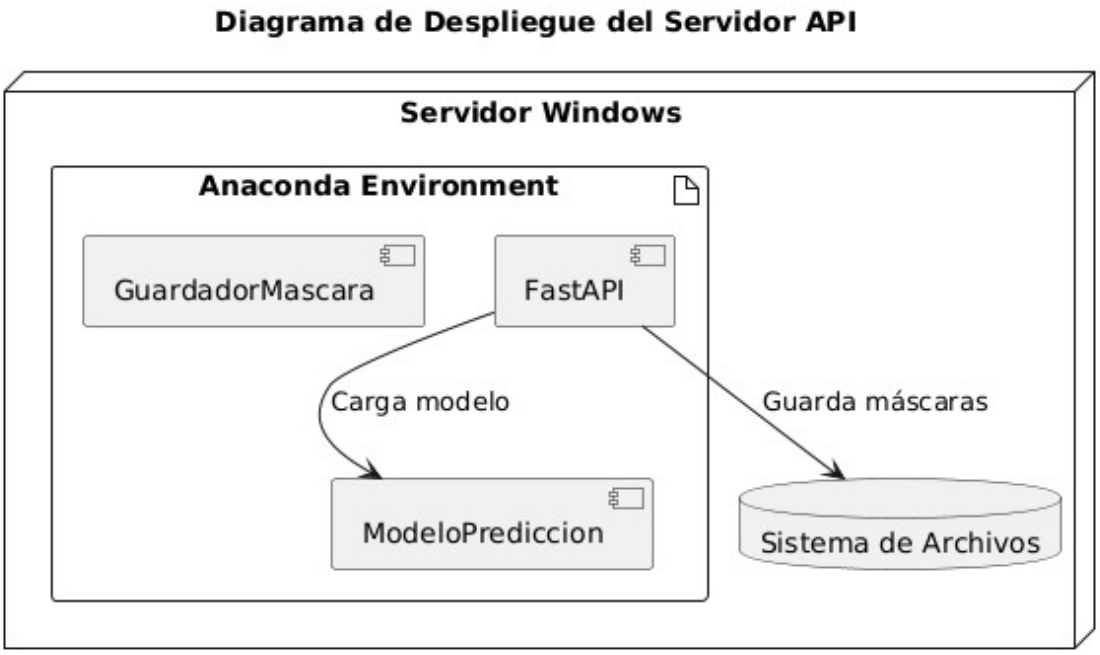

Figura 29: Flujograma servidor

Diagrama de Estados

El diagrama de estados fue elaborado para mostrar los diferentes estados por los que pasa una imagen dentro del sistema. Esto incluyó estados como 'recibido', 'preprocesado', 'predicción', y 'máscara generada', brindando un control detallado sobre el flujo del sistema y posibles puntos de fallo.

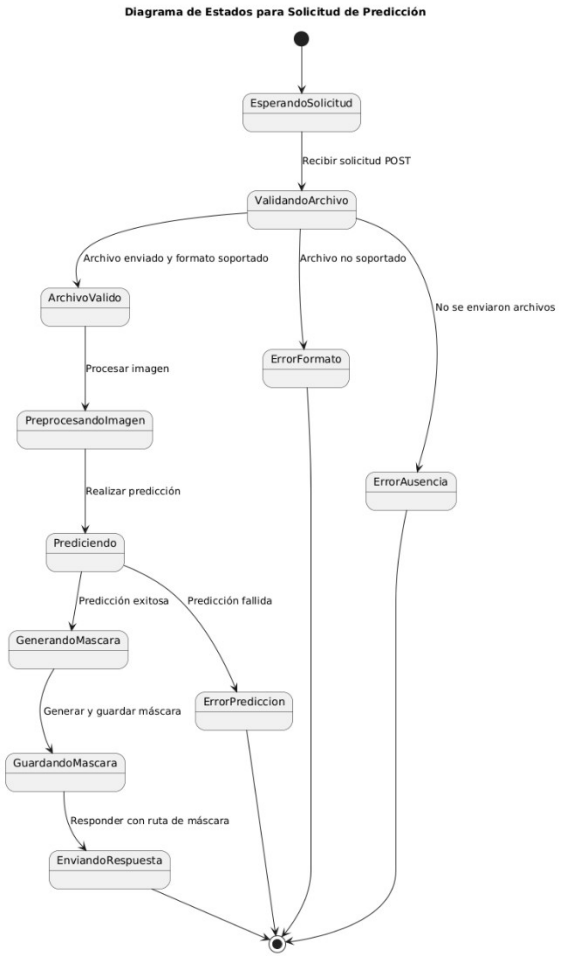

Figura 30: Flujograma petición de predicciones

El servidor API fue capaz de procesar imágenes en tiempo real, proporcionando predicciones precisas tanto para cuerpos de agua como de bosque. La solución manejó múltiples solicitudes concurrentes de manera eficiente, devolviendo imágenes segmentadas en formato PNG que destacaban las áreas detectadas de interés.

DIAGRAMA FINAL

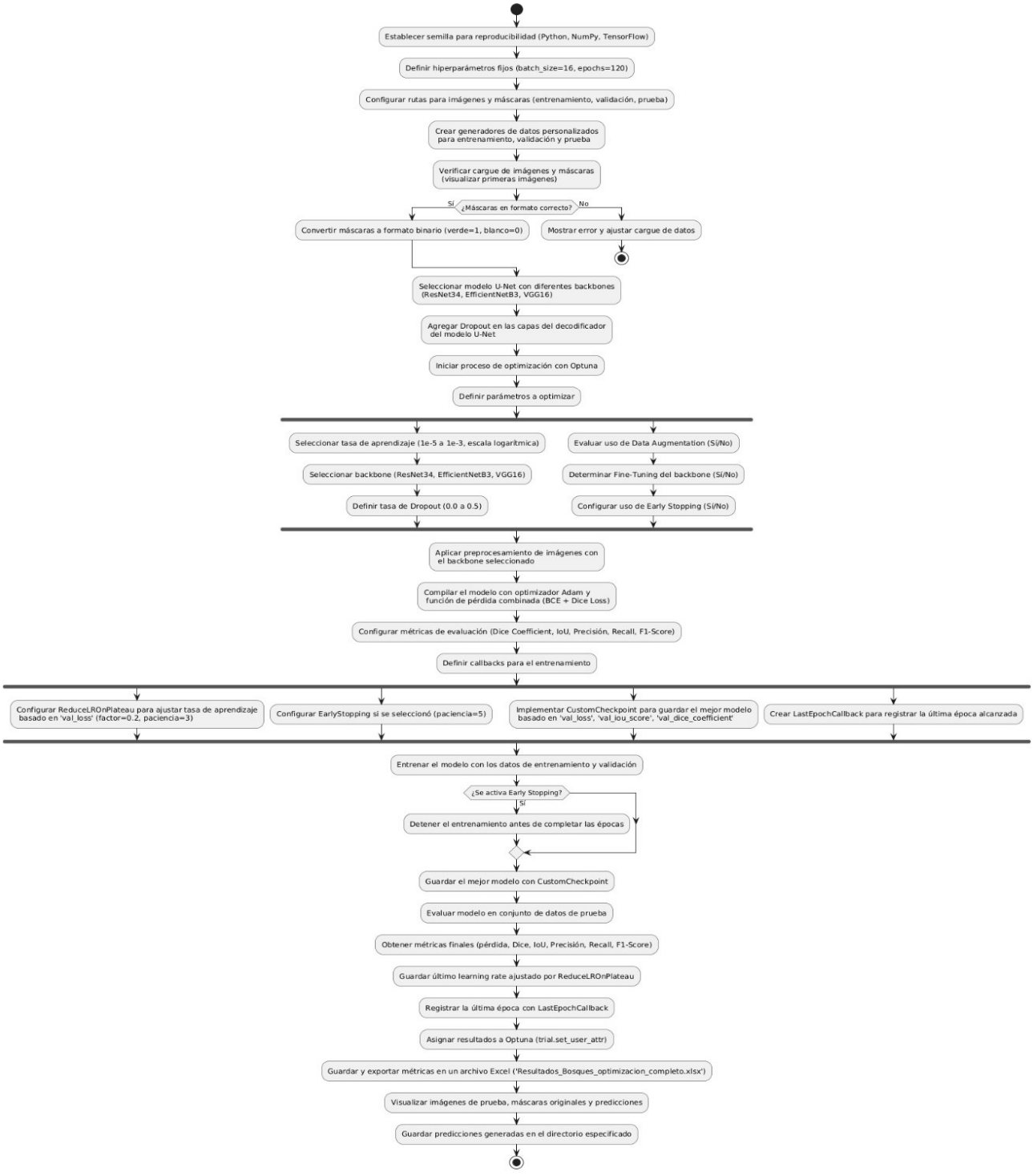

Figura 31: Flujograma general del modelo

8. CONCLUSIONES Y TRABAJOS FUTUROS

8.1 CONCLUSIONES

El problema que abordamos consistía en la necesidad de desarrollar un sistema que, basado en imágenes satelitales o aéreas, pudiera identificar y segmentar cuerpos de agua y bosques. Esta solución debía estar orientada a procesar imágenes en un servidor de manera eficiente, brindando resultados de predicción en tiempo real mediante el uso de modelos de redes neuronales convolucionales (CNNs). Nuestro análisis inicial se enfocó en entender el flujo de datos que implicaba la recepción de imágenes por parte de un cliente, el posterior preprocesamiento de esas imágenes y la ejecución de predicciones mediante un modelo de TensorFlow ya entrenado. Adicionalmente, debíamos generar y devolver imágenes de segmentación, conocidas como máscaras, que resaltaran los cuerpos de agua o bosques presentes en las imágenes de entrada.

Decidimos utilizar **FastAPI** para crear el servidor API debido a su eficiencia en la gestión de solicitudes **HTTP** y su capacidad para integrar modelos de aprendizaje profundo. Además, su estructura asíncrona permitía gestionar múltiples solicitudes de manera eficiente. Para la predicción de imágenes, utilizamos **TensorFlow** con modelos entrenados previamente en la segmentación de cuerpos de agua y bosques. **OpenCV** se seleccionó como la herramienta para procesar las imágenes y generar las máscaras, mientras que **Pillow** nos permitió manejar la lectura y preprocesamiento de las imágenes.

1. **Definición de las métricas y funciones personalizadas**

Iniciamos definiendo las funciones de pérdida y métricas personalizadas que utilizamos en el entrenamiento de los modelos de segmentación. Estas funciones incluyeron el coeficiente de Dice, pérdida combinada (BCE y Dice), precisión, recall, F1-Score e IoU.

2. **Creación del servidor API con FastAPI**

Implementamos un **servidor API** con **FastAPI** que recibía imágenes a través de peticiones HTTP y

procesaba dichas imágenes para realizar la predicción. El servidor exponía dos rutas principales: **/predecir/agua** para cuerpos de agua y **/predecir/bosque** para cuerpos de bosque.

3. Preprocesamiento de las imágenes

Para que los modelos de predicción pudieran trabajar con las imágenes enviadas, desarrollamos una función de preprocesamiento que incluía la lectura, redimensionado y normalización, asegurando así la compatibilidad con los modelos.

4. Carga de los modelos de TensorFlow

Cargamos dos modelos de redes neuronales convolucionales (CNNs) previamente entrenados, uno para la segmentación de cuerpos de agua y otro para bosques, utilizando custom_object_scope para incluir las métricas personalizadas definidas anteriormente.

5. Generación y guardado de las máscaras

Después de realizar la predicción, generamos máscaras que resaltaban las áreas segmentadas, utilizando OpenCV para colorear las áreas de interés y guardarlas como archivos PNG que se devolvían al cliente.

6. Manejo de errores

Implementamos un sistema de manejo de errores para gestionar situaciones como formatos de imagen no soportados, fallos en la predicción y ausencia de archivos.

4. Implementación de la solución

La implementación se realizó en un entorno de Windows utilizando Anaconda para gestionar las dependencias. El servidor fue desplegado con **Uvicorn**, permitiendo gestionar solicitudes concurrentes de manera eficiente.

8.2 TRABAJOS FUTUROS

Si bien nos centramos en U-Net con estos backbones preentrenados, consideramos que los otros modelos que analizamos, como DeepLabV3+ y FPN (Feature Pyramid Network), podrían ser utilizados en un futuro proyecto para mejoras o pruebas adicionales. Estos modelos más avanzados podrían ofrecer beneficios en escenarios con más complejidad o cuando se requiera un análisis más detallado y como posibles opciones para mejorar el rendimiento general de la segmentación o explorar nuevas técnicas en análisis posteriores.

También dentro del ámbito de los mercados verdes, este proyecto es el prólogo a una herramienta que se está buscando establecer, la cual es una calculadora de carbono automatizada, la cual sería

la automatización del siguiente paso del en un proyecto de bonos de carbono, que es determinar la cantidad de carbono que se puede secuestrar, dependiendo de la cantidad de biomasa leñosa y otros factores claves para el proyecto.

Luego, referente al modelo de bosques , el modelo actual basado en EfficientNetB3 es sumamente eficiente, pero para mejorar su capacidad de generalización y adaptabilidad a nuevos entornos, se recomienda la introducción de data augmentation, fine-tuning del backbone, y la exploración de arquitecturas más profundas o híbridas. Además, se podrían utilizar imágenes de mayor resolución y funciones de pérdida especializadas para capturar más detalles en áreas complejas. Finalmente, evaluar el modelo en diferentes entornos y aplicar transfer learning avanzado podría ser clave para optimizar aún más su rendimiento y utilidad en diversas aplicaciones de segmentación forestal.

Por otro lado, en el caso del modelo de aguas, basado en EfficientNetB3 ha mostrado ser muy eficaz en la segmentación de cuerpos de agua, pero su capacidad de generalización puede mejorar mediante la incorporación de data augmentation, fine-tuning del backbone, y el uso de funciones de pérdida avanzadas. Asimismo, probar el modelo en imágenes de mayor resolución y con una mayor diversidad de tipos de cuerpos de agua permitiría una mejor adaptación a diferentes contextos geográficos y climáticos. Finalmente, realizar un fine-tuning progresivo y ajustar el modelo para datos más variados sería crucial para asegurar su eficacia en aplicaciones prácticas más complejas.

9. REFERENCIAS BIBLIOGRÁFICAS

[1] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 1ª ed. Sebastopol, CA: O'Reilly Media, 2019.

[2] H. Ferdous, T. Siraj, S. J. Setu, M. M. Anwar, y M. A. Rahman, "Machine Learning Approach Towards Satellite Image Classification," 2020.

[3] M. K. Kolli, C. Opp, D. Karthe, y M. Groll, "Mapping of Major Land-Use Changes in the Kolleru Lake Freshwater Ecosystem by Using Landsat Satellite Images in Google Earth Engine," 2020.

[4] N. B. Toosi, A. R. Soffianian, S. Fakheran, S. Pourmanafi, C. Ginzler, y L. T. Waser, "Land Cover Classification in Mangrove Ecosystems Based on VHR Satellite Data and Machine Learning—An Upscaling Approach," 2020.

[5] C. Ünsalan y K. L. Boyer, *Multispectral Satellite Image Understanding: From Land Classification to Building and Road Detection*, 1ª ed. Dordrecht: Springer, 2017.

[6] T. Acharya y A. K. Ray, *Image Processing: Principles and Applications*, 1ª ed. London: Springer, 2005.

[7] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1ª ed. New York: Springer, 2006.

[8] D. Michie, D. J. Spiegelhalter, y C. C. Taylor, *Machine Learning, Neural and Statistical Classification*, 1ª ed. Chichester: Wiley, 1994.

[9] N. J. Nilsson, *Introduction to Machine Learning*, 1ª ed. USA: Stanford University, 1996.

[10] S. Behnke, *Hierarchical Neural Networks for Image Interpretation*, 1ª ed. Berlin: Springer, 2003.

[11] J. Jorgenson, *Ecosystem Ecology*, 1ª ed. London: Academic Press, 2005.

[12] I. Goodfellow, Y. Bengio, y A. Courville, *Deep Learning*, 1ª ed. Cambridge, MA: MIT Press, 2016.

[13] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, 1ª ed. Cambridge, MA: MIT Press, 2012.

[14] T. Hastie, R. Tibshirani, y J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2ª ed. New York: Springer, 2009.