UNIVERSIDAD NACIONAL DE INGENIERÍA
FACULTAD DE CIENCIAS

TESIS

"SEGMENTACIÓN SEMÁNTICA DE VÍAS EN IMÁGENES SATELITALES DE ALTA RESOLUCIÓN MEDIANTE REDES CONVOLUCIONALES"

PARA OBTENER EL GRADO ACADÉMICO DE MAESTRO EN CIENCIAS EN CIENCIA DE LA COMPUTACIÓN CON MENCIÓN EN ESPECIALIDAD COMPUTACIÓN CIENTÍFICA

ELABORADO POR

EDSON GREIG CÁCERES CÁRDENAS

ASESOR

DR. CÉSAR ARMANDO BELTRÁN CASTAÑÓN

LIMA – PERÚ

2024

DEDICATORIA

A mi compañera de vida, Eunice, por el soporte y compresión que siempre me ha otorgado en cada paso que he dado para cumplir mis objetivos personales.

A mi hijo, Lucas, esperando que, al leer este mensaje, haya encontrado el propósito en su vida (if not just keep going), y que nuestras charlas sobre ciencia hayan contribuido en ello.

A la más maravillosa madre que Dios pudo haber puesto en mi camino, Laura. No solo por el apoyo incondicional, sino también por haber sembrado en mí, el gusto de aprender y superarme, y en consecuencia hacer mi camino en lo académico más llevadero.

AGRADECIMIENTOS

Al Dr. César Armando Beltrán Castañón, por su constante paciencia y buen ánimo, pero sobre todo por su valiosa guía y asesoramiento desde la concepción hasta las etapas finales de este trabajo de investigación.

A la Facultad de Ciencias de la Universidad Nacional de Ingeniería, por haber brindado media beca en la Maestría en Ciencia de la Computación, no solo a mi persona sino a toda mi promoción de ingreso, lo cual hizo más sencillo acceder a educación de calidad.

RESUMEN

La segmentación semántica de vías en imágenes satelitales es un campo muy importante y estudiado en el estado del arte, ya que, disponer de la infraestructura vial es bastante significativo para la toma de decisiones en diversas áreas de un país. La recuperación de esta información se da con grandes procesos que consumen tiempo considerable y que pueden involucrar gran desplazamiento logístico. Por lo anterior, es importante contar con un método el cual nos permita disponer de manera rápida y automática esta información. Este trabajo tuvo como objetivo desarrollar un modelo basado en redes convolucionales para la segmentación multiclase automática de vías en imágenes satelitales. Al no disponer de un conjunto de datos para optimizar nuestro modelo, se define un procedimiento a fin de crear muestras a partir de un conjunto de datos base. La evaluación cualitativa exhibe una apropiada generalización del modelo de segmentación, ya que, las inferencias del modelo sobresalen sobre las máscaras de las imágenes en los experimentos. De igual manera, las evaluaciones cuantitativas coinciden con las cualitativas, obteniendo valores por encima del 93.97% en todas las métricas. Finalmente, al no haber trabajos que aborden el problema de manera multiclase, se espera que esta investigación sirva como marco de referencia para futuros trabajos.

Palabras claves: Deep Learning, redes convolucionales, segmentación semántica multiclase, infraestructura vial, imágenes satelitales.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ACRÓNIMOS

CNN	Convolutional Neural Network
IEEE	Institute of Electrical and Electronics Engineers
IBM	International Business Machine Corporation
ACM	Association for Computing Machinery
ANN	Artificial Neural Network
ViT	Vision Transformer
MLP	Multi-Layer Perceptron
RGB	Red Green Blue
MPP	Marked Point Process
ResNet	Residual Network
IoU	Interception over Union
SVM	Super Vector Machine
HR	High Resolution
VHR	Very High Resolution
UNETR	UNEt TRansformer

CAPÍTULO I: INTRODUCCIÓN

En este primer capítulo, se exponen las motivaciones para el desarrollo del presente trabajo de investigación orientadas al ámbito académico y social. Asimismo, se plantean los objetivos principales y específicos. Finalmente, se indica los aportes del presente trabajo a la comunidad científica y a la sociedad.

1.1. Motivación

La información de la infraestructura vial es vital para una correcta gestión a nivel de estado, gobiernos regionales y municipales en muchos aspectos, ya sea, en gestión de obras o toma de decisiones para determinados eventos y/o desastres. Según [1], se conoce que del territorio nacional de vías aproximadamente 30 208 km está pavimentado y cerca de 143 402 km no lo están. Estas cifras actualmente podrían estar obsoletas, y es que este tipo de información es tan dinámica, que en el transcurso de los días y/o semanas podría variar, sin mencionar que la recolección de esta información suele ser muy complicada por parte de los tres niveles de gobierno, esta se podría recuperar mediante visitas a campo o mediante algún procesamiento manual en un software de computadora. Disponer de un mecanismo que nos permita visualizar la infraestructura vial de una determinada zona de manera rápida y automática sería muy productivo para cualquiera de los tres niveles de gobierno, ya que, con esto se eliminarían los procedimientos para la recuperación de la información y solamente quedaría procesar la información de la infraestructura vial para eventualmente decidir sobre esta.

La segmentación semántica de vías en imágenes satelitales o aéreas es un campo bastante estudiado en el estado del arte. Aunque aparentemente la detección de estos objetos puede parecer sencilla, ya que, la morfología es simple a comparación de otras (un edificio, por ejemplo), esta no es una tarea fácil, ya que, estos objetos pueden variar en color, forma, anchura, etc. La literatura deja ver alternativas de abordar este problema en el marco del procesamiento digital de imágenes, soluciones que van desde algoritmos muy bien elaborados hasta la construcción de modelo computacionales basados en *Deep Learning*. Actualmente los métodos más populares de segmentación están basados en redes

convolucionales (CNN, por sus siglas en inglés), las cual es una solución que aprende de la misma imagen, y es capaz de reconocer diferentes patrones en cada nivel de la red.

La presente tesis se enfoca en el procedimiento de segmentación de vías en imágenes satelitales de alta resolución. A lo largo de los años han sido propuestos varios modelos de segmentación semántica [2], [3], [4]. La finalidad de la presente investigación es desarrollar un modelo que automáticamente aprenda a extraer características de los píxeles de las imágenes satelitales y utilizar las características extraídas para determinar las clases predefinidas.

1.2. Objetivos

1.2.1. Objetivo general

Desarrollar un modelo basado en redes convolucionales para la segmentación multiclase automática de vías en imágenes satelitales.

1.2.2. Objetivos específicos

- Definir un procedimiento para preprocesar y construir una base de datos de imágenes para las fases de entrenamiento, validación y prueba.
- Definir las métricas apropiadas para medir la calidad de segmentación.
- Implementar un modelo convolucional para la segmentación multiclase de vías en imágenes satelitales.
- Evaluar el desempeño del modelo a nivel de la calidad de segmentación.

1.3. Justificación

1.3.1. Justificación técnica

En el estado del arte se ha podido evidenciar que hay diversos enfoques para la segmentación de vías, todos y cada uno de estos abordan el problema desde un punto de vista binario, es decir, solo se considera el objeto de interés (la vía) y el fondo (todos los demás objetos). Ninguno de los autores ha abordado el problema desde una perspectiva multiclase, de tal forma que se disponga de más información de la infraestructura vial como resultado de la segmentación. Por lo cual, este trabajo busca abordar el problema de segmentación multiclase de vías de imágenes satelitales a fin de proponer un modelo computacional que lo resuelva de manera automática.

1.3.2. Justificación social

Como se dijo anteriormente, la infraestructura vial es un factor muy importante en la gestión de los diferentes tipos de gobiernos y entidades. Disponer de la información vial proporcionada por una imagen satelital de una determinada ciudad puede contribuir a la toma de decisiones sobre problemas relacionados a la planificación y mantenimiento urbano y vial, actualización de mapas, e incluso en esta última se podría detectar aquellas zonas que estarían más aisladas (sin muchas vías de acceso). Actualmente, las organizaciones que tienen la necesidad de esta información recurren a recolectar información en campo o también a aplicar propiamente la segmentación de imágenes. En esta última, se suelen aplicar procedimientos semiautomáticos orientados al procesamiento digital de imágenes y software especializados. El problema de estos métodos radica en el hecho que depende de la intervención y subjetividad del usuario, también que algunos de estos softwares pueden estar sujetos a licencias de pago, y además que estos procedimientos pueden tener un tiempo de procesamiento considerable. Por ello, es sumamente significativo contar con un método con el cual podamos disponer oportunamente de la información de la infraestructura vial, y que este pueda ser replicado y quizás adaptado a ciertas circunstancias.

1.4. Alcance

En el presente trabajo de investigación se elabora el diseño, construcción, optimización y evaluación de un modelo computacional basado en redes convolucionales para realizar la segmentación semántica de vías sobre imágenes satelitales de alta resolución. Para ello se tiene el siguiente alcance:

- El trabajo busca explorar y aplicar redes convolucionales para segmentar vías en imágenes satelitales. Las etiquetas definidas son fondo, vía pavimentada y vía no pavimentada; por lo cual se debe considerar un conjunto de datos que contenga estas tres etiquetas.

1.5. Limitaciones

A continuación, se lista las limitaciones en el desarrollo de la presente tesis:

- No se dispone de un conjunto de datos que aborde el problema de segmentación considerando las tres etiquetas definidas previamente.

- No existe marco de referencia con el cual contrastar eventuales resultados obtenidos.

1.6. Aportes

Al no haberse abordado por la comunidad científica el problema de segmentación multiclase de vías, no se dispone de muestras para entrenar y evaluar alguna solución propuesta. Por esto, una contribución importante es un procedimiento, basado en procesamiento digital de imágenes digitales, que nos permita apropiadamente obtener un conjunto de datos para poder diseñar un eventual método de solución para la comunidad. Importante también mencionar que dar a conocer nuevas estrategias de generar conjuntos de entrenamiento ayuda como base para otras áreas en la teledetección en las cuales no se cuenta con gran cantidad de datos de entrenamiento. Asimismo, al no haberse tocado la segmentación de vías de manera multiclase, se crea un nuevo marco de referencia, el cual puede servir de comparación para futuros trabajos de investigación.

Por otro lado, otorgar a la comunidad una arquitectura que permita disponer de la información de la arquitectura vial de manera oportuna es valioso para las organizaciones que requieren este tipo de información, contribuyendo al proceso de recolección de información en estas instituciones.

Finalmente, constatar la viabilidad del aprendizaje profundo a la segmentación multiclase de vías podría generar nuevas y más complejas aplicaciones relacionadas.

CAPÍTULO II: MARCO TEÓRICO

En este capítulo se discute los principales conceptos relacionados al tema de esta investigación, tales como segmentación de imágenes, técnicas de segmentación, hasta conceptos dentro del marco del *Deep Learning*.

2.1. Visión por computador

Ciencia que permite que las maquinas vean, aquí ver en este caso significa que la máquina es capaz de extraer información de una imagen que es necesaria para resolver alguna tarea [5]. En visión por computador intentamos describir el mundo que vemos en una o más imágenes y reconstruir sus propiedades, como la forma, la iluminación y la distribución del color [6].

2.2. Segmentación de imágenes

La segmentación de imágenes divide una imagen en regiones, llamadas segmentos, para propósitos de un mayor análisis de la imagen, mejorar la eficiencia de la compresión de la imagen, o simplemente para efectos de visualización. Matemáticamente, dividimos la imagen I en un número finito de segmentos, tal que:

- $S_i \neq \emptyset, para\ toda\ i \in \{1, \dots, n\}$.
- $\bigcup_{i=1}^{n} S_i = I$.
- $S_i \cap S_j = \emptyset, para\ todo\ i, j \in \{1, \dots, n\}\ donde\ i \neq j$.

La segmentación de imágenes crea segmentos de píxeles conectados al analizar algún criterio de similitud, posiblemente soportado por la detección de píxeles que muestran alguna diferencia con la adyacencia de píxeles. La segmentación apunta a identificar segmentos "significativos" que pueden ser usados para describir el contenido de una imagen [7].

2.3. Técnicas de segmentación de imágenes

Existe un buen número de algoritmos presentes en la literatura, ninguno de estos métodos funcionaria para toda imagen, dependería del tipo de imagen, algunas de estas técnicas son:

2.3.1. Basadas en umbral

[7] sugiere aplicar a menudo solamente un umbral global T para mapear una imagen I en una imagen binaria:

$$J(x,y) = \begin{cases} 0, & if\ I(x,y) < T \\ 1, & otherwise \end{cases}$$

El umbral global (T) puede ser identificado por una optimización estratégica apuntando a crear regiones conectadas largas y reducir el número de regiones de tamaño menor [7].

2.3.2. Basadas en detección de bordes

El método de detección de borde es comúnmente un método usado para detectar discontinuidad en una imagen. Es una manera popular de detectar píxeles de bordes y relacionarlos para crear bordes en una imagen, los píxeles de bordes son píxeles en los cuales hay una transición en valor de intensidad [8]. Uno de los más populares dentro de este grupo es la detección de bordes basadas en gradientes.

Detección de bordes basado en gradiente: Es comúnmente usado en procesamiento de imágenes para encontrar la primera derivada de una imagen. La gradiente es un vector de dos dimensiones que apunta a la dirección en el cual la intensidad de la imagen crece más rápido. Las dos funciones que pueden ser expresadas en términos de derivadas direccionales son la magnitud y la orientación de la gradiente. La gradiente es definida por [8]:

$$g(x,y) = \sqrt{\Delta x^2 + \Delta y^2}$$

Donde:

$$n : distancia\ más\ pequeña\ entre\ ubicaciones\ de\ los\ píxeles$$

$$\Delta x = f(x+n, y) - f(x-n, y)$$

$$\Delta y = f(x, y+n) - f(x, y-n)$$

Esta cantidad da el máximo ratio de crecimiento de $f(x, y)$ por unidad de distancia en la orientación de la gradiente $g(x, y)$. La orientación de la gradiente está definida por [8]:

$$\emptyset(x, y) = \text{atan}\left({}^{\Delta y}/_{\Delta x}\right)$$

2.3.3. Basados en clustering

Esta técnica de agrupamiento intenta acceder a las relaciones entre los patrones del conjunto de datos organizando los patrones en grupos o conglomerados de manera que los patrones dentro de un conglomerado sean más similares entre sí que los patrones que pertenecen a diferentes conglomerados. Es decir, el agrupamiento se refiere a la clasificación de objetos en grupos según ciertas propiedades de estos objetos. Un procedimiento estándar para la agrupación es asignar cada píxel a la clase de la media de agrupación más cercana [9]. El agrupamiento de K-Means es un método popular de agrupamiento duro que divide una imagen en k agrupamientos, el algoritmo es el siguiente [8]:

1. Seleccione la intensidad de k píxeles como centroide inicial.
2. Forme k grupos asignando todos los píxeles al centroide más cercano.
3. Vuelva a calcular el centroide de cada grupo tomando la media de los valores de intensidad de píxeles dentro del grupo y reasignando los píxeles.
4. Repita el mismo proceso hasta que el centroide no cambie más.

2.4. Segmentación semántica

La segmentación de imágenes puede ser formulada como un problema de clasificación de píxeles con etiquetas semánticas. La segmentación semántica realiza etiquetados a nivel de píxel con un conjunto de categorías (humano, carro, árbol, cielo) para todas las imágenes [10]. La tarea es agrupar partes de una imagen juntas las cuales pertenecen al mismo objeto. Este tipo de algoritmos tiene varios casos de uso tales como detectar señales viales, detectar tumores, detectar instrumentos médicos en operaciones, clasificación del uso y cobertura del suelo [11]. La siguiente imagen nos da una idea visual de la segmentación semántica [12]:

Figura 1. Segmentación semántica

2.5. Inteligencia artificial

El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) define la inteligencia artificial como la teoría y el desarrollo de sistemas informáticos que sean capaces de realizar tareas que normalmente requieren inteligencia humana, como la percepción visual, el reconocimiento del habla, el aprendizaje, la toma de decisiones y el procesamiento del lenguaje natural [13].

Inteligencia artificial término utilizado para sistemas computacionales que intentan imitar aspectos de la inteligencia humana, incluidas funciones que asociamos intuitivamente con la inteligencia, como el aprendizaje, la resolución de problemas y el pensamiento y la acción racional [14].

2.6. Machine Learning

Arthur Samuel, define Machine Learning como el campo de estudio que da a las computadoras la habilitad de aprender sin ser explícitamente programadas.

Para la Corporación Internacional de Maquinas de Negocios (IBM) es el uso y desarrollo de sistemas computacionales que son capaces de aprender y adaptarse sin seguir instrucciones explícitas, mediante el uso de algoritmos y modelos estadísticos para analizar y sacar inferencias a partir de patrones en los datos [15].

La Asociación de Maquinaria de Computación (ACM) define Machine Learning (o aprendizaje de máquinas) como la ciencia que enseña a las computadoras cómo descubrir, contextualizar y actuar con precisión sobre los datos a través de la experiencia [16].

Machine Learning se define como un conjunto de métodos que pueden detectar automáticamente patrones en los datos y luego usar los patrones descubiertos para predecir datos futuros o para realizar otros tipos de toma de decisiones en condiciones de

incertidumbre (como planificar cómo recopilar más datos). Este campo puede ser dividido en tres tipos [17]:

- **Aprendizaje supervisado:** El objetivo es aprender a mapear entradas x a salidas y, dado un conjunto etiquetado de pares $D = \{(x_i, y_i)\}_{i=1}^{N}$. D es llamado el conjunto de entrenamiento, y N es el numero de muestras de entrenamiento
- **Aprendizaje no supervisado:** En este tipo, solo disponemos de las entradas, $D = \{x_i\}_{i=1}^{N}$, y el objetivo es encontrar patrones interesantes en la data. A esto a veces se le llama descubrimiento de conocimiento. Este es un problema mucho menos bien definido, ya que no se nos dice qué tipo de patrones buscar y no existe una métrica de error obvia para usar (a diferencia del aprendizaje supervisado, donde podemos comparar nuestra predicción de y para una x dada con el valor observado).
- **Aprendizaje por refuerzo**: Existe un tercer tipo de aprendizaje automático, conocido como aprendizaje por refuerzo, que se utiliza con menor frecuencia. Esto es útil para aprender cómo actuar o comportarse cuando se le dan señales ocasionales de recompensa o castigo.

2.7. Deep Learning

Deep Learning (o aprendizaje profundo) permite a los modelos computaciones que son compuestos de múltiples capas de procesamiento aprender la representación de la información con múltiples niveles de abstracción. Estos métodos han mejorado dramáticamente el estado del arte en reconocimiento de voz, reconocimiento de objetos, detección de objetos y muchos otros dominios. *Deep Learning* descubre estructuras intrincadas en grandes conjuntos de datos mediante el uso del algoritmo de retropropagación para indicar cómo una máquina debe cambiar sus parámetros internos que se utilizan para calcular la representación en cada capa a partir de la representación en la capa anterior [18]. Las redes convolucionales, el cual es un tipo de red neuronal dentro del *Deep Learning*, son un ejemplo del valioso aporte de este campo en la visión por computador.

Un algoritmo de aprendizaje profundo extrae automáticamente las características de alto y bajo nivel necesarias para la clasificación. Por características de alto nivel, se entiende una característica que depende jerárquicamente de otras características. Por ejemplo, en el contexto de la visión por computadora, esto implica que un algoritmo de aprendizaje

profundo aprenderá sus propias representaciones de bajo nivel a partir de una imagen sin procesar (como un detector de bordes, filtros Gabor, etc.) y luego construirá representaciones que dependen de esas representaciones de bajo nivel (como combinaciones lineales o no lineales de esas representaciones de bajo nivel) y repetir sucesivamente el mismo proceso para niveles superiores [19].

2.8. Redes neuronales artificiales

Las redes neuronales artificiales (ANN, por sus siglas en inglés) se definen como un sistema de mapeos no lineales cuya estructura se basa en principios observados en los sistemas nerviosos de humanos y animales [20]. Las redes neuronales están compuestas de nodos o unidades (ver Figura 2) conectadas a través de conexiones dirigidas. Una conexión de la unidad j a la unidad i sirve para propagar la activcion a_j de j a i [21].

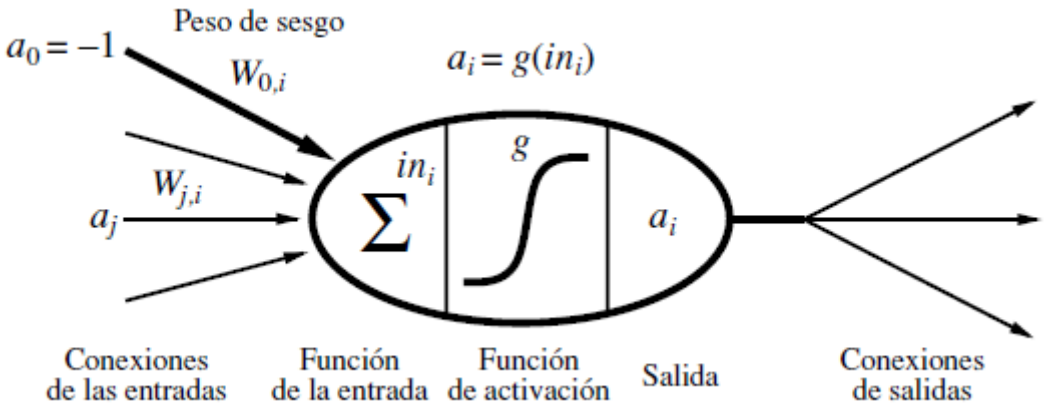


Figura 2. Modelo matemático para una neurona

Fuente: Russel y Norving [21]

Cada conexión tiene un peso numérico $W_{j,i}$ asociado, que determina la fuerza y el signo de la conexión. Cada unidad i primero calcula la suma ponderada de sus entradas [21]:

$$in_i = \sum_{j=0}^{n} W_{j,i} a_j$$

Luego aplica la función de activación g a esta suma para producir la salida [21]:

$$a_i = g(in_i) = g(\sum_{j=0}^{n} W_{j,i} a_j)$$

2.9. Redes neuronales convolucionales

Las redes convolucionales también conocidas como redes neuronales convolucionales o CNN, son un tipo especializado de red neuronal para procesar datos que tiene una topología conocida en forma de cuadrícula. Los ejemplos incluyen datos de series temporales, que pueden considerarse como una cuadrícula 1D que toma muestras a intervalos de tiempo regulares, y datos de imágenes, que pueden considerarse como una cuadrícula 2D de píxeles. Las redes convolucionales han tenido un gran éxito en aplicaciones prácticas. El nombre "red neuronal convolucional" indica que la red emplea una operación matemática llamada convolución. La convolución es un tipo especializado de operación lineal. Las redes convolucionales son simplemente redes neuronales que utilizan convolución en lugar de la multiplicación general de matrices en al menos una de sus capas [22].

2.9.1. Arquitectura de las redes convolucionales

[23] define la arquitectura de las redes convolucionales en el siguiente grafico:

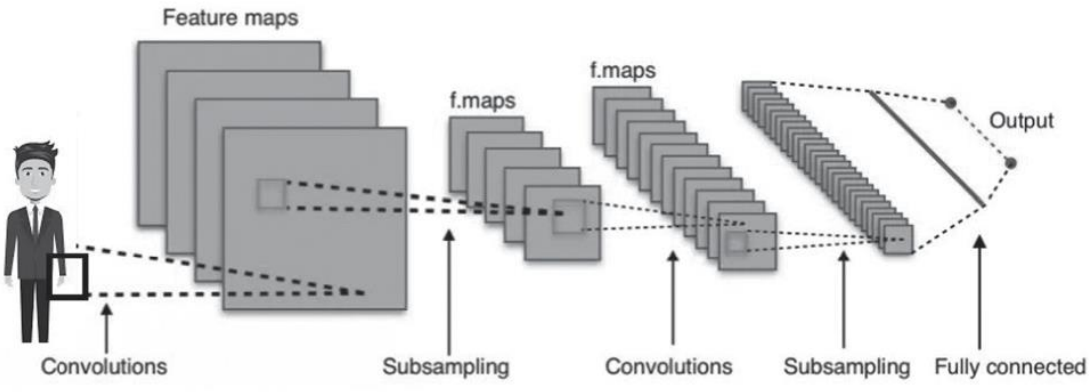


Figura 3. Arquitectura - Red convolucional

- **Capa convolucional:** este es el proceso central. Aquí, un filtro pasará (convolucionará) sobre la imagen de entrada, escaneará los píxeles y luego creará un mapa de características. Cada característica pertenece a alguna clase. El filtro esencialmente pasa por encima de la imagen de entrada y este proceso se denomina convolución (también tenemos que incluir funciones de activación en esta capa).
- **Capa *pooling* (*Down sampling/sub sampling*):** es el proceso de reducir la cantidad de píxeles (es decir, muestreo descendente) sin perder la información

importante. Un método que discutimos en detalle fue la agrupación máxima en la que podíamos retener los píxeles más fuertes e ignorar los más débiles. Se debe tener en cuenta que puede haber muchas rondas de convolución y agrupación.

- ***Flattening*:** La salida de las capas anteriores se aplana en un solo vector para que puedan ingresar al siguiente nivel. Antes de entregar los resultados a las capas completamente conectadas después de la convolución, es importante aplanarlos para que puedan ser procesados por las capas completamente conectadas.
- **Capa totalmente conectada (*fully connected*):** Esta capa toma las entradas del análisis de características y aplica pesos para predecir la etiqueta correcta.
- **Capa de salida (*output*):** Esto nos da las probabilidades finales para cada etiqueta. Aquí es donde se puede obtener el resultado final.

2.10. Auto atención

El núcleo de un enfoque basado en la atención es la capacidad de comparar un elemento de interés con una colección de otros elementos de una manera que muestre su relevancia en el contexto actual. En el caso de la autoatención, el conjunto de comparaciones es con otros elementos dentro de una secuencia determinada [24].

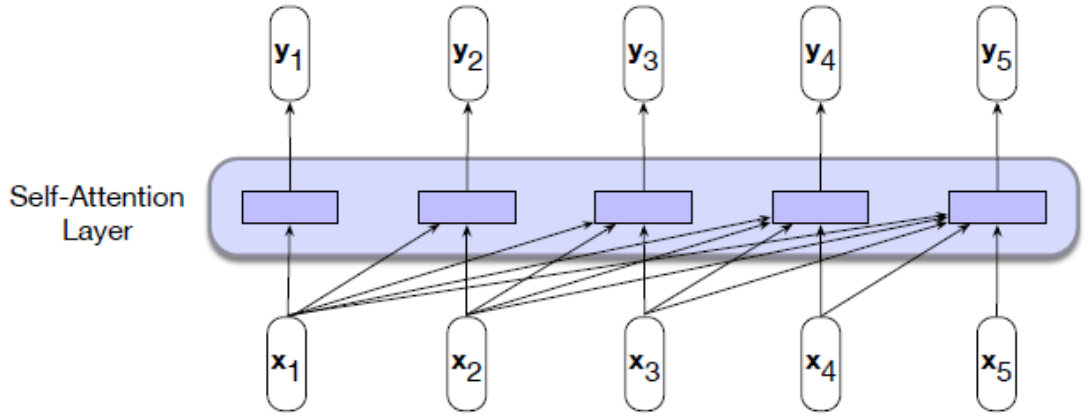


Figura 4. Mecanismo de atención

Fuente: Jurafsky y Martin [24]

El resultado de estas comparaciones se utiliza luego para calcular una salida para la entrada actual. Por ejemplo, en la Figura 4, el cálculo de y_3 se basa en un conjunto de comparaciones entre la entrada x_3 y sus elementos precedentes x_1 y x_2, y con x_3 mismo. La

forma más sencilla de comparación entre elementos de una capa de autoatención es un producto escalar. Nos referiremos al resultado de esta comparación como una puntuación [24]:

$$score(x_i, x_j) = x_i \,.\, x_j$$

Luego, para hacer un uso eficaz de estas puntuaciones, las normalizaremos con un *softmax* para crear un vector de pesos, α_{ij}, que indica la relevancia proporcional de cada entrada para el elemento de entrada i que es el foco de atención actual [24].

$$\alpha_{ij} = \text{softmax}\left(score(x_i, x_j)\right) \quad \forall\, j \leq i$$

Dadas las puntuaciones proporcionales en α, generamos un valor de salida y_i tomando la suma de las entradas vistas hasta ahora, ponderadas por su respectivo valor α [24].

$$y_i = \sum_{j \leq i} \alpha_{ij} x_j$$

2.11. Vision Transformer

Vision Transformer (ViT) es un modelo propuesto originalmente por [25], para aplicar los Transformers, utilizados en tareas de procesamiento de lenguaje natural, y con este la auto atención. El siguiente modelo, ideado originalmente para clasificación de imágenes que emplea una arquitectura de *Transformers* sobre parches de imágenes.

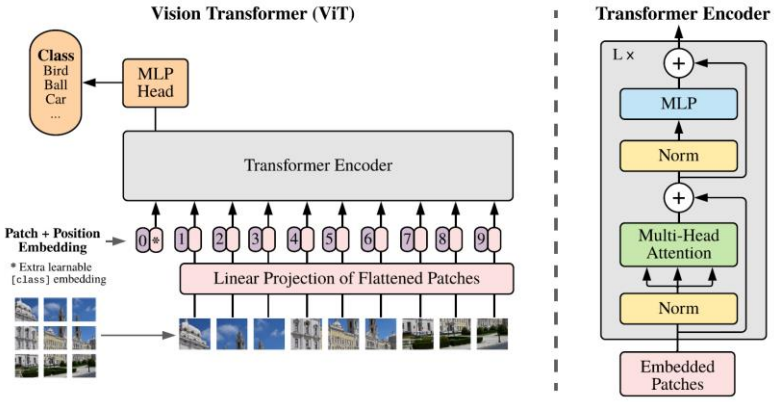


Figura 5: Descripción general de Vision Transformers

Fuente: Dosovitskiy, Beyer, Kolesnikov, Weissenborn, Zhai, Unterthiner, Dehghani, Minderer, Heigold, Gelly, Uszkoreit y Houlsby [25]

Como se puede observar en la Figura 5, el enfoque sugiere que una imagen sea dividida en parches de tamaño fijo, luego cada uno de ellas se incrusta linealmente, se agregan incrustaciones de posición y la secuencia resultante de vectores se envía a un codificador *Transformer* estándar. Al final del modelo propuesto, la salida del *Transformer* es la entrada de un *Muli Layer Perceptron* (MLP), con el fin de poder inferir la etiqueta sobre la imagen de entrada.

2.12. Infraestructura vial

La infraestructura vial viene a ser lo que constituye la vía y todos sus soportes que conforman la estructura de carreteras y caminos [26].

Por infraestructura vial se entiende al conjunto de medios técnicos, servicios e instalaciones que componen la vía pública, necesarios para el tránsito de personas y objetos en forma segura y confortable desde un punto a otro [27].

CAPÍTULO III: ESTADO DEL ARTE

En el presente capítulo, se realiza una descripción de los trabajos relacionados a segmentación de imágenes satelitales. Estos trabajos nos dan una orientación de cuan estudiado es este campo, así como también nos sirve como punto de partida para el desarrollo del presente trabajo de investigación. Esta sección se divide en métodos basados en procesamiento de imágenes digitales tradicional y métodos basados en aprendizaje de máquinas.

3.1. Métodos basados en procesamiento de imágenes digitales tradicional

[28] plantea un método de extracción de información de imágenes orientada a objetos. Primero, y con fines de que la imagen sea más adecuada para la siguiente fase, esta es pasada por un proceso de mejora de la imagen a través de un estiramiento lineal de 2%. Posteriormente, se utilizan algoritmos de segmentación basado en bordes e intensidad, los cuales en esencia son algoritmos basados en umbrales. Finalmente, se lleva a cabo la extracción de la vía, mediante ciertas características del objeto, tales como, el área, ya que las vías suelen ser superficies largas y estrechas con áreas considerables. La sucesión de estos pasos logra la clasificación de la vía en la imagen de entrada, aunque el método suele ser bastante sensible a la calidad de la imagen y a la información que trae en ella.

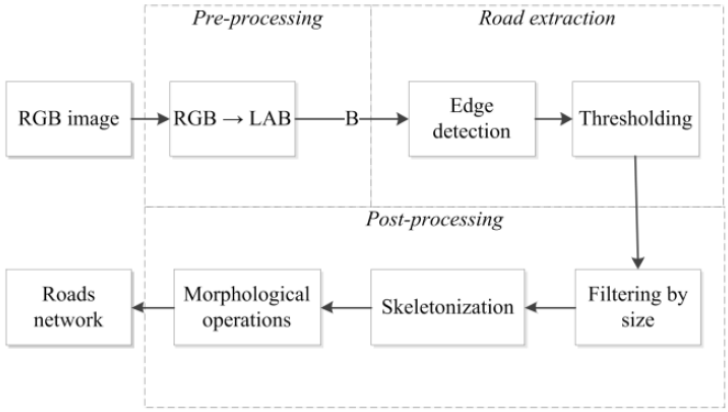


Figura 6: Esquema general - Método propuesto por Kurbatova y Laylina

Fuente: Kurbatova y Laylina [29]

Por otro lado, [29] propone un enfoque que utiliza el método de segmentación de bordes basado en las cadenas de Markov. Este enfoque se divide en tres fases tales como el preprocesamiento, la extracción de la vía y el posprocesamiento, como muestra en Figura 6. En el preprocesamiento, las imágenes RGB son convertidas al espacio de colores Lab debido a que su componente B los valores son más invariantes. En la segunda fase, se segmenta el contorno con el método basado en las cadenas de Markov de dos dimensiones para la extracción de la vía. Finalmente, la vía extraída es sometida a un posprocesamiento con el fin de mejorar la calidad de este. Los experimentos realizados en este trabajo confirman que las operaciones morfológicas, de filtrado, de esqueletización mejoran la calidad de la imagen.

Por su parte, [30] propone un método basado únicamente en algoritmos de detección de bordes y operaciones morfológicas para extraer la red de vías de una imagen satelital multiespectral de alta resolución. Las imágenes son pasadas por los algoritmos de detección de bordes tales como *canny*, *prewitt* y *sobel*; finalmente, el filtro mediana es aplicado para reducir el ruido presente en las imágenes. La calidad de las imágenes finales obtenidas por el método es moderada, incluso al usar el filtro mencionado, ya que, en algunos casos, pequeñas partes de tierra y áreas de estacionamiento son clasificados como carreteras**.**

De igual forma, [31] propone una técnica la cual puede ser resumida en cuatro pasos. De las imágenes RGB, el canal B es tomado, debido a su mejor contraste, para ser pasado por un filtro mediana con el fin de mejorar la calidad de la imagen. Luego, un algoritmo de segmentación, el cual es llamado Active Contour Model, con este se obtiene la morfología de las vías presentes en la imagen. Posteriormente, las imágenes segmentadas son sometidas a un proceso de posprocesamiento, a fin de remover el ruido presente. Finalmente, se aplica la distancia euclidiana para reducir el número de componentes conectados en la imagen obtenida en el paso previo y para mostrar la línea central de la carretera. La calidad de las imágenes obtenidas por la propuesta es moderada, por lo que el mismo autor sugiere mejoras para utilizar el modelo en imágenes de mucha más resolución.

[32] desarrolla una técnica basada en objetos de varias etapas. Primero, las imágenes satelitales son suavizadas utilizando un filtro guiado, el cual mejora la calidad de los resultados obtenidos en la fase de segmentación multiresolución. Después de la segmentación, cada píxel es asignado a un determinado objeto. Con el fin de diferenciarlos de las vías, las características de estos objetos (vías) son tomados en cuenta, algunas de ellas

son: las vías son estructuras largas y con anchos acotados, no son áreas pequeñas y son regiones homogéneas. Posteriormente, un algoritmo de tensor de votación personalizado llena las partes faltantes en las redes de las vías, y finalmente, las imágenes pasan por un proceso de vectorización y poda el cual remueve regiones colindantes o tomadas en cuenta como la red de vías. La propuesta es evaluada sobre tres criterios (completitud, corrección y calidad), los resultados sobre estos criterios siempre se encuentran entre los dos mejores en todos los diferentes datasets utilizados.

Cambiar a un diferente espacio de colores es algo a tener en cuenta en cuanto a análisis de color de objetos, como inicialmente considera [29]. [33] posterior al cambio de espacio de colores a Lab, se obtiene la matriz de distancias D del conjunto de entrenamiento, con el fin de realizar una segmentación en las imágenes. Después, el algoritmo de Otsu es aplicado, con el cual se obtienen las máscaras de las vías presentes en las imágenes. Finalmente, para obtener los bordes de las vías, se aplica el algoritmo de detección de bordes *Canny* sobre las máscaras obtenidas en el paso previo. Los experimentos realizados indican practicidad en el enfoque, y que este puede ser aplicado para detectar incluso otros objetos sobre imágenes satelitales.

3.2. Métodos basados en aprendizaje de máquinas

[34] propone todo un marco de trabajo, descrito en la Figura 7, el cual empieza por una fase de segmentación basada en una red neuronal de la familia U-Net. Posteriormente, los resultados del modelo neuronal son la entrada del modelo Connected-Tube MPP, propuesta del mismo autor, a fin de extraer con precisión las carreteras como tubos conectados. Finalmente, los resultados obtenidos hasta ese momento son pasados por un algoritmo de rastreo de vías, para completar o llenar aquellas partes en donde se evidencia una discontinuidad. Los dos pasos previos mejoran los problemas de conectividad presentados en los resultados iniciales, solucionado problemas de conectividad en las vías, produciendo mejores resultados finales.

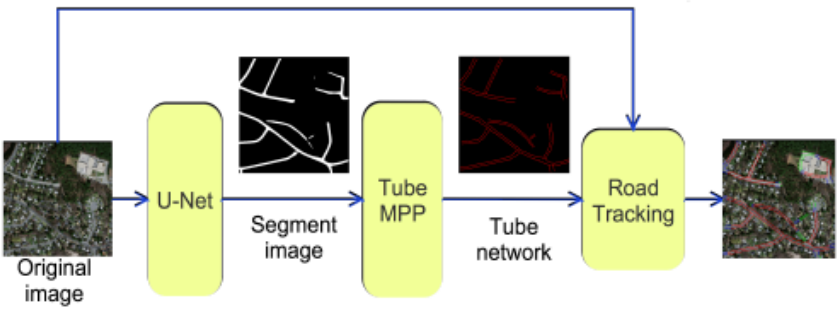


Figura 7: Esquema general - Método propuesto por Li, Comer y Zerubia

Fuente: Li, Comer y Zerubia [34]

Por su parte, [35] propone un enfoque basado en mejoras de las arquitecturas U-Net y ResNet, llamado U-Net y ResNeXt, el cual considera bloques residuales en este, donde las capas aprenden de los residuos de estos bloques. Los mapas semánticos predichos por este modelo presentan ruido, debido a esto, las predicciones pasan por un posprocesamiento, el cual está básicamente basado en un procesamiento de imágenes digitales, todo esto contribuye a que la máscara resultante sea muy pulcra en la definición de sus vías. Los resultados fueron comparados con modelos entrenados sobre el mismo dataset, y se muestra que el modelo propuesto supera en 3% sobre la puntuación F1.

[36] propone un método para la segmentación de imágenes satelitales de alta resolución que se puede visualizar en Figura 8. Antes de pasar la imagen por el bloque de la red que realiza la segmentación, la imagen es pasada por un *encoder*, el cual codifica las características en alta resolución, el cual ayuda a los mapas de atención, posteriormente la imagen es pasada por la sección de la red encargada de producir la máscara segmentada de la imagen. La propuesta considera pasar el resultado obtenido (mapa de segmentación) por una sección adicional de red neuronal, el cual detecta los bordes de las vías, el cual es el producto final del modelo. Los resultados obtenidos se equiparán con el modelo U-Net comparado sobre las métricas IoU y *Dice Score*.

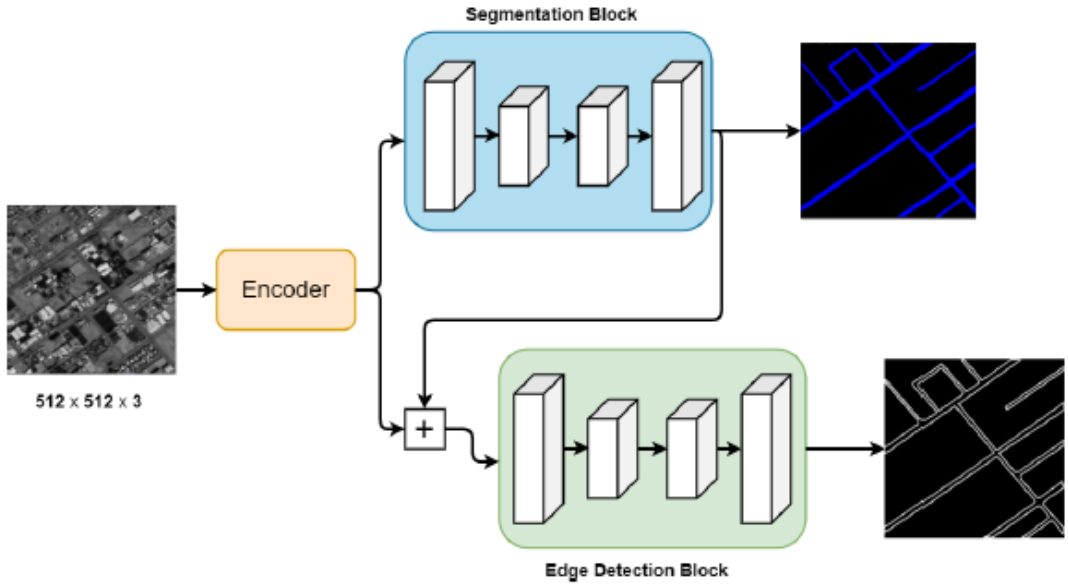


Figura 8: Esquema general - Método propuesto por Ghandorh, Boulila, Masood, Koubaa, Ahmed y Ahmad

Fuente: Ghandorh, Boulila, Masood, Koubaa, Ahmed y Ahmad [36]

El uso del *Transfer Learning* es regularmente considerado en problemas de clasificación de píxeles. [37], [38] y [34] proponen técnicas de segmentación de imágenes satelitales basado en un modelo pre-entrenado, generalmente estos ajustados sobre ImageNet. Para [37], primero, las imágenes satelitales son, evidentemente, de gran dimensión espacial, por lo que lo hacen difícil para el entrenamiento en un sistema de memoria y recursos computacionales limitados. Por lo que, el total de cada imagen satelital es dividida en sub-imágenes. Segundo, los recortes son pasados por un algoritmo de *clustering* de píxeles basados en la proximidad del color. Finalmente, todo esto hace que las imágenes resultantes sean más sencillas de segmentar en el modelo pre-entrenado. Los experimentos realizados muestran valores por encima de 90%, sobre las métricas evaluadas.

También es posible aplicar métodos de clasificación usados en el aprendizaje automático para segmentar imágenes, tales sean como *Random* Forest o Super Vector Machine (SVM), por ejemplo, [39] y [40] sugieren cada uno un enfoque en el cual se considera el SVM. [39] personaliza el uso de este clasificador en cuatro fases. Inicialmente, un detector de bordes *canny* es empleado para segmentar las vías en la imagen. Después, se aplica el método de fusión Full Lambda Schedule para combinar segmentos adyacentes. Como tercer paso, SVM es aplicado para clasificar los píxeles de la imagen. Finalmente, operaciones morfológicas como la dilatación, erosión, *opening* y *closing* son aplicadas a las

imágenes con el fin de remover los objetos no deseados. Según [39], la precisión general y la media del coeficiente kappa obtenidos indican un éxito general del método propuesto para la extracción de vías.

Por otro lado, [41] propone un enfoque basado en la destilación del conocimiento, en donde se realiza la transferencia de conocimiento de una arquitectura se transfiere a otra. La propuesta tiene dos arquitecturas U-Nets, la primera trabaja sobre imágenes satelitales de alta resolución (HR), y la otra trabaja sobre imágenes satelitales de muy alta resolución (VHR). Adicionalmente, las imágenes VHR que carecen de etiqueta pasan por la primera arquitectura, con el cual se obtiene su máscara correspondiente, el cual es utilizada para entrenar la segunda arquitectura U-Net. En los experimentos realizados, se evidencia que el rendimiento general es similar en ambos modelos, aunque el segundo modelo supera ligeramente en casos en que las imágenes satelitales representan escenarios más complejos.

Finalmente, [42] demuestran que su enfoque puede beneficiar significativamente la extracción de vías al aprovechar la información de orientación de la vía. El enfoque consta de tres componentes, como se aprecia en Figura 9: un codificador, una subred de mejoramiento de características iterativas y un decodificador multitareas. Adoptan una arquitectura ResNet con cuatro bloques residuales como codificador de características, el cual toma como entrada una imagen RGB y devuelve un mapa de características de un cuarto del tamaño original. Como corazón de la red está la subred de mejora de características iterativa, el cual consiste en una cascada de pilas de mejoramiento de características. Cada pila primera obtiene características para extracción de la vía y predicción de orientación, respectivamente, luego se realiza mejoramiento de características usando los módulos propuestos. Como paso final, el decodificador produce la extracción de vías y resultados de orientación usando las correspondientes características mejoradas.

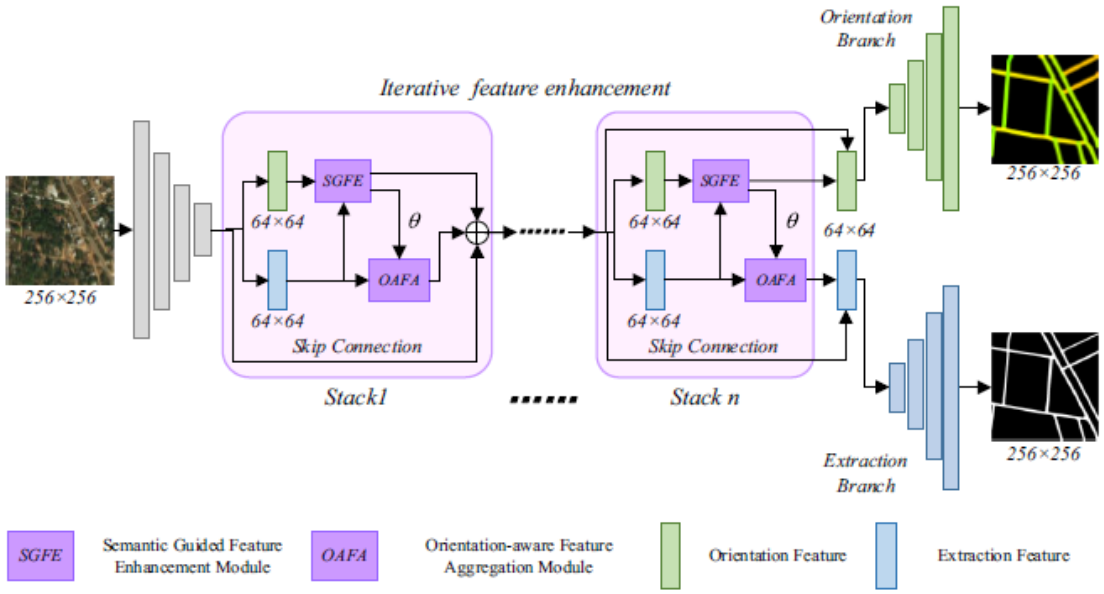


Figura 9: Esquema general - Método propuesto por Yin, Qian, Wang, Qi y Lu

Fuente: Yin, Qian, Wang, Qi y Lu [42]

CAPÍTULO IV: METODOLOGÍA E IMPLEMENTACIÓN

En este capítulo definimos la metodología usada para la implementación del presente trabajo y se detallan las fases más importantes durante el desarrollo del mismo. Adicionalmente, se detalla las herramientas disponibles para este trabajo y la implementación propia donde se detalla un poco más el esquema general presentado.

4.1. Esquema general

Con el propósito de alcanzar los objetivos planteados, se propone una metodología simple, la cual se puede apreciar en la Figura 10, en esta se muestra tres grandes fases en las cuales está dividida la implementación del trabajo, estas son:

- Elaboración del dataset
- Construcción y entrenamiento del modelo
- Evaluación del modelo

4.1.1. Elaboración del dataset

Evidentemente para poder entrenar algún modelo de *Deep Learning*, requerimos los datos visuales que se utilizarán en la segunda fase. Al no estar a disposición ningún dataset que aborde el problema de segmentación de vías de manera multiclase, corresponde elaborar uno propio. Considerando DeepGlobe [43] como dataset base, se procede a realizar una serie de pasos sucesivos, detalladas en Figura 10, con el fin de poder construir nuestro dataset para poder entrenar a nuestro modelo de segmentación.

4.1.2. Construcción y entrenamiento del modelo

Una vez que se puede disponer de un conjunto considerable de datos, corresponde la construcción del modelo. Se ha iniciado por primero implementar el modelo de segmentación, a fin de que se sepa las características de entrada de este para poder realizar el preprocesamiento adecuado al dataset. Después, al ya disponer de nuestros datos para entrenamiento y el modelo implementado, se procede a realizar la distribución del dataset

en conjuntos de entrenamiento, validación y prueba; todo esto con el motivo de poder realizar el ajuste del modelo.

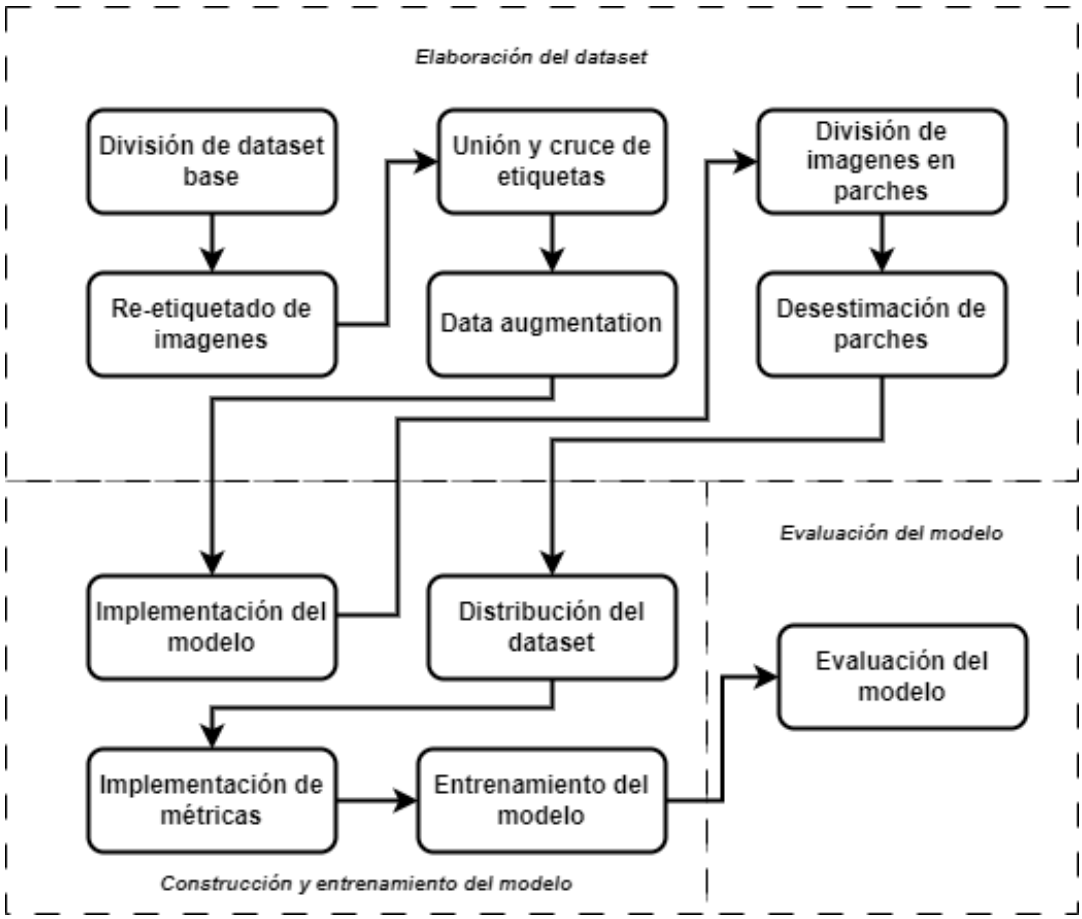


Figura 10. Metodología de trabajo

4.1.3. Evaluación

Finalmente, una vez que el modelo ha sido ajustado, corresponde evaluar el performance del modelo, teniendo en cuenta las métricas más idóneas para un problema de segmentación. El proceso de evaluar el modelo sugiere la comparación del mapa de segmentación obtenido por el modelo propuesto junto con la máscara (*ground truth*) de la imagen de entrada

4.2. Herramientas utilizadas

Para el presente trabajo de investigación se ha utilizado un conjunto de recursos divididos en dos grupos principales: hardware y software.

4.2.1. Hardware

Para poder realizar cada uno de los pasos detallados en la metodología se dispuso de un equipo con las siguientes características:

- Memoria RAM: 128 GB
- Almacenamiento: 2 TB SDD
- Procesador: Intel Core i9 11900KF 3.5GHZ 11thGeneration, 16 Nucleos
- Tarjeta gráfica: NVIDIA RTX 3090

4.2.2. Software

Para el procesamiento de imágenes digitales y la implementación del modelo de segmentación se utilizaron los siguientes recursos:

- Sistema operativo: Ubuntu 20.04.5 LTS
- *Docker* versión 20.10.21
- *CUDA* versión 11.4.
- Lenguaje de programación *Python* versión 3.8.10.
- Librería *Tensorflow* versión 2.10.0.
- Librería *OpenCV* versión 4.6.0.
- Software *Label Studio* 1.6.0.

4.3. Implementación del trabajo

La Figura 10 resume la metodología implementada en el presente trabajo, en esta sección se procede a especificar a más detalle los pasos realizados.

4.3.1. División del dataset base

Se toma como base el dataset DeepGlobe [43], las características de este dataset son las siguientes:

Tabla 1. Características DeepGlobe Dataset

Total de imágenes utilizadas	**6226**
Dimensiones	1024x1024
Etiquetas	0: Background 1: Vía
Canales	RGB
Resolución del satélite	50 cm/pixel

Lo limitante de este dataset es que solo considera el problema de detección de vías de manera binaria, es decir, el valor 0 corresponde al fondo (background) y 1 corresponde a una vía. Este dataset no puede ser tomado como tal para nuestro propósito (considerar el problema de manera multiclase), es por ello que se procede a separar manualmente las imágenes en tres grupos, los cuales son pavimentados, no pavimentadas y ambas (imágenes que tienen en ella los dos tipos de vía).

Tabla 2. Resultado de división manual del dataset base

Grupo	Número de imágenes
Pavimentadas (paved)	3150
No pavimentadas (not paved)	967
Ambas (both)	2109
Total	**6226**

4.3.2. Re-etiquetado de imágenes satelitales

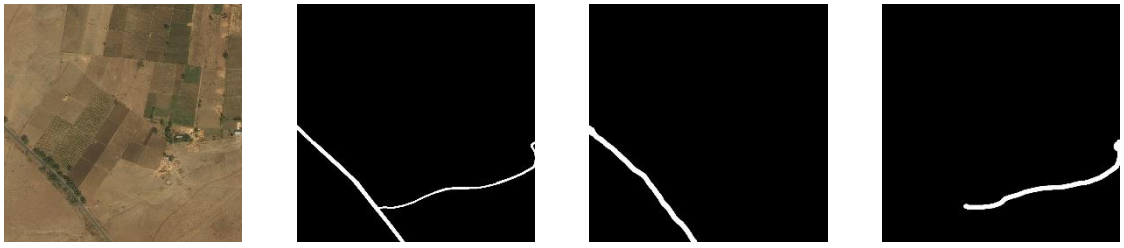
De la Tabla 2, se puede apreciar un grupo de imágenes (pavimentadas y no pavimentadas) que estarían listas para ser consideras en un siguiente paso, pero hay un grupo de imágenes que tienen que ser tratadas de como un caso especial, que son las denominadas "ambas", éstas son las imágenes que en su contenido disponen tanto de vías pavimentadas y

no pavimentadas, y aun no podrían ser consideradas para el preprocesamiento de la imagen porque no se sabría que etiqueta asignar a los píxeles con valor 1.

Con el propósito de aprovechar las imágenes categorizadas como ambas, se hace uso del software *Label Studio* para realizar el etiquetado manual de estas imágenes y poder generar etiquetas para su uso.

4.3.3. Unión y cruce de etiquetas

Después de haber realizado el etiquetado de las imágenes en el *Label Studio*, se obtienen imágenes de la siguiente forma:

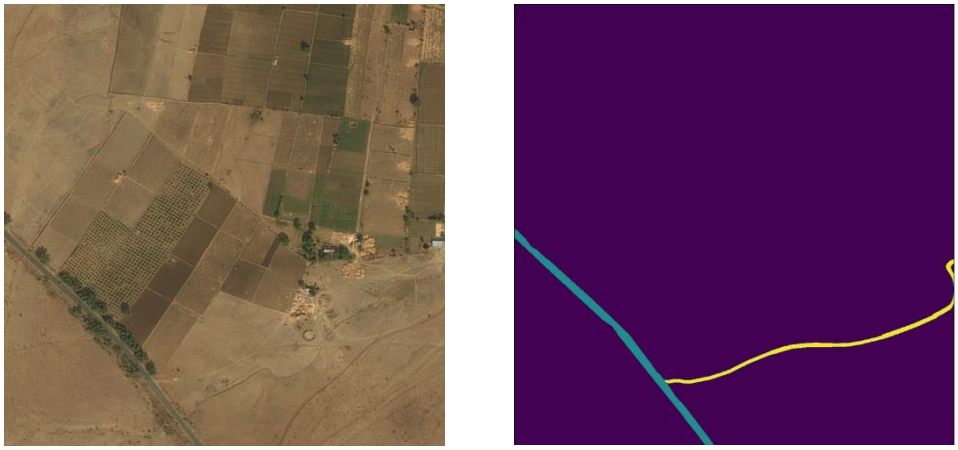


a b c d

Figura 11. Muestra de etiquetado a) Imagen original b) Etiqueta binaria original c) Etiqueta pavimentada (binaria) d) Etiqueta no pavimentada (binaria)

En la Figura 11, se puede apreciar (en c y d) que *Label Studio* genera una imagen para cada etiqueta que haya en la imagen, y como en la imagen se ha procedido a etiquetar dos veces (una para la vía pavimentada y otra para la no pavimentada), se ha generado dos etiquetas, ambas binarias. Si bien se podría trabajar con imágenes de este tipo, pero por comodidad se procede a aplicar procesamiento de imágenes con el fin de unir las etiquetas que corresponden a una imagen, así solo obtendremos una etiqueta por imagen.

Aprovechando el procesamiento de la imagen, también se procede a hacer un cruce entre las etiquetas generadas y la etiqueta original, con el fin de mejorar la calidad de nuestra etiqueta final, la Figura 12 muestra la etiqueta final:



a b

Figura 12. Resultado de etiquetado final. a) Imagen original b) Etiqueta final no binaria (morado: fondo, celeste: pavimentado, amarillo: no pavimentado)

De esta forma se puede disponer del total de imágenes de la Tabla 2 para la construcción de nuestro dataset.

4.3.4. Data augmentation

La mejor manera para hacer que un modelo de *Machine Learning* generalice mejor es entrenarlo con más datos. Por supuesto, en la práctica, la cantidad de datos que tenemos es limitada. Una forma de evitar este problema es crear datos no reales [22].

Asimismo, en la Tabla 2 se puede apreciar un desbalance en los dos primeros grupos en nuestro dataset base, con el fin de también balancear estos grupos tanto como se pueda se aplica la generación de datos sintéticos sobre nuestro dataset base, algunas de las transformaciones sobre las imágenes fueron flips horizontal, flips vertical, rotaciones, etc., el resultado de este proceso se muestra en la Tabla 3:

Tabla 3. Resultados de la generación de datos sintéticos

Grupo	Etiqueta	Número de imágenes
Pavimentadas	paved	3150
No pavimentadas	not paved	967
Both	paved y not paved	2109
Pavimentadas aug.	paved	6300
No pavimentadas aug.	not paved	8703
Total	**-**	**21 229**

Tabla 4. Dataset imágenes 1024x1024

Grupo	Etiqueta	Número de imágenes
Pavimentadas	paved	9450
No pavimentadas	not paved	9670
Both	paved y not paved	2109
Total	**-**	**21 229**

De esta forma, se consigue incrementar el número de nuestro dataset base considerablemente a la par que se busca balancear la distribución de etiquetas en número de imágenes. En la Tabla 4, vemos que el desbalance se ha removido, y que la diferencia de imágenes entre la etiqueta paved y not paved es mínima (sin considerar el grupo both).

4.3.5. Implementación de modelos

Proponemos dos arquitecturas, basadas en U-Net. Hemos escogido estos tipos de arquitectura porque son representativas del estado del arte en tareas de segmentación, pero más importante, eficientes en su desempeño y capaces de trabajar con grades cantidades de datos.

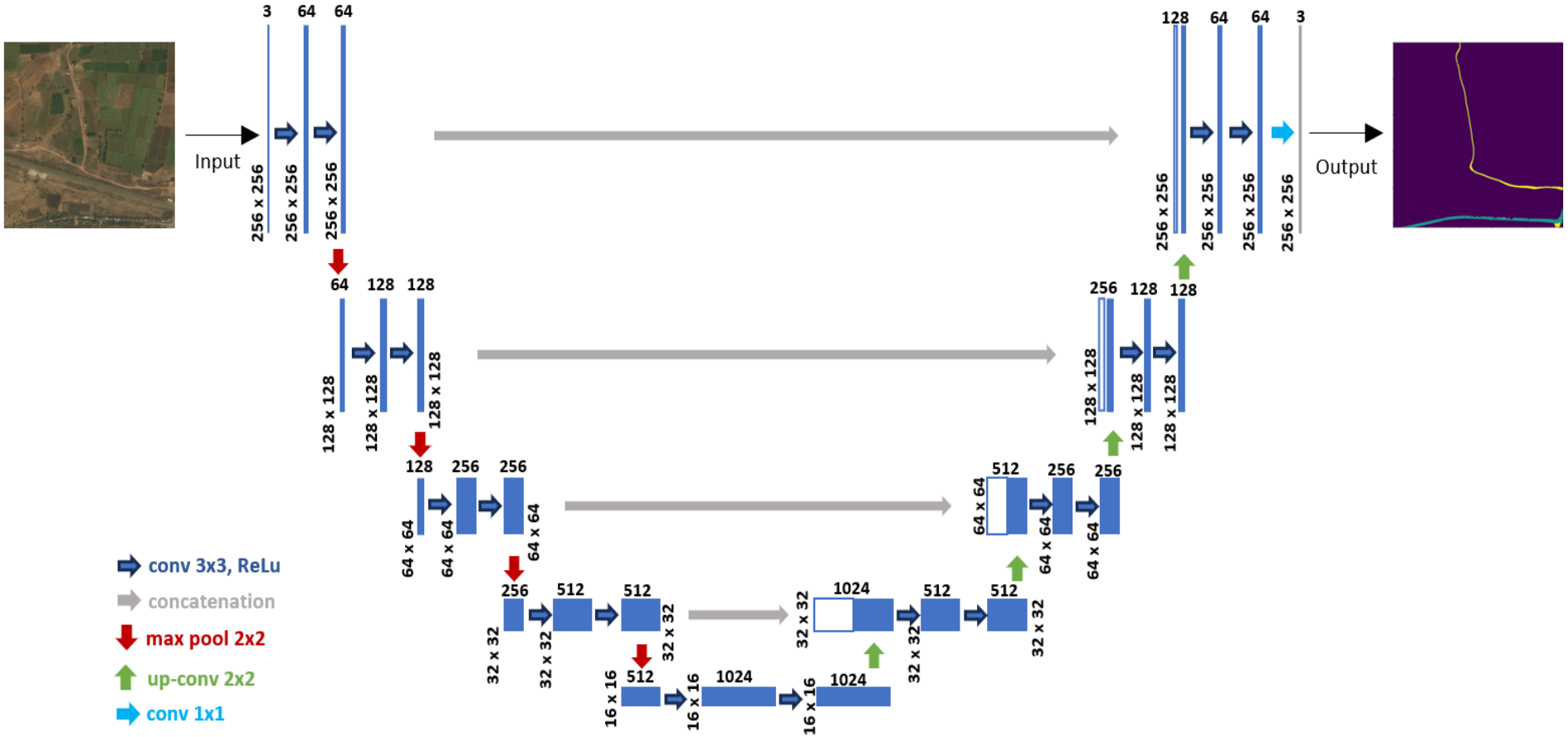


Figura 13. Arquitectura U-Net

4.3.5.1. Arquitectura U-Net

Para la primera red, nos basamos en U-Net [4] pero con ligeras modificaciones sobre la estructura propuesta originalmente. Este modelo, representado en Figura 13, consiste de una ruta de contracción (lado izquierdo) y una ruta de expansión (lado derecho). La ruta de contracción evidentemente muestra la arquitectura común de una red convolucional, el cual consiste en repetidas aplicaciones de dos convoluciones de 3x3, a cada una de estas convoluciones se aplica una función de activación ReLu, una capa *dropout* y una capa de normalización en lotes. Posterior a esto se aplica una operación *max pooling* (con tamaño de ventana de 2x2 y *stride* de 2) para reducir las dimensiones espaciales del mapa de características, a la vez que se duplica el número de canales de características.

En la ruta de expansión se llevan a cabo repetidas operaciones de deconvoluciones (o *upsampling*) sobre el mapa de características, operación que reduce a la mitad el número de canales de características, esta reducción será compensada con la concatenación con el correspondiente mapa de características de la ruta de contracción. Al resultado obtenido hasta aquí se le aplica dos convoluciones, de las mismas características aplicadas en la ruta de contracción.

En la parte final de la arquitectura, se agrega una convolución de 1x1 para mapear los 64 mapas de características a nuestro número de clases, las cuales son tres.

4.3.5.2. Arquitectura UNETR

La segunda arquitectura llamada UNETR, basada en [44] para imágenes 3D, fue inspirada en U-Net [4], por lo que trabaja similar con una ruta de contracción (codificador) y un camino de expansión (decodificador).

Figura 14 muestra lo innovador de esta arquitectura, el cual utiliza *ViT*, propuesto originalmente por [25], como codificador de las características de imágenes. Y el decodificador es uno basado en una red convolucional, como es usual. Los *Transformers* operan sobre una secuencia 1D de *embeddings* de entrada. Por lo que creamos una secuencia 1D de una imagen 2D, $x \in \mathbb{R}^{256x256x3}$, dividiendo cada imagen en un conjunto de parches aplanados, considerando cada parche con dimensiones 16x16, se obtendria $x_p \in \mathbb{R}^{256x768}$. Posteriormente, una capa lineal es usada para proyectar los parches en un espacio de *embeddings* de K dimensiones. Luego, un embedding posicional es aplicado para guardar la

información espacial de los parches. Después, los parches son codificados por el *Transformer*, a la salida final (Z_{12}) se le aplica una capa de deconvolución con el fin de incrementar su resolución. Seguido, se concatena el mapa de característica redimensionado con el mapa de características de la salida anterior del Transformer (Z_9), y una convolución de 3x3 y una deconvolución son aplicadas. Se ha de repetir el proceso hasta la salida final, en donde una convolución 1x1 es aplicada para mapear los 64 mapas de características al número de etiquetas.

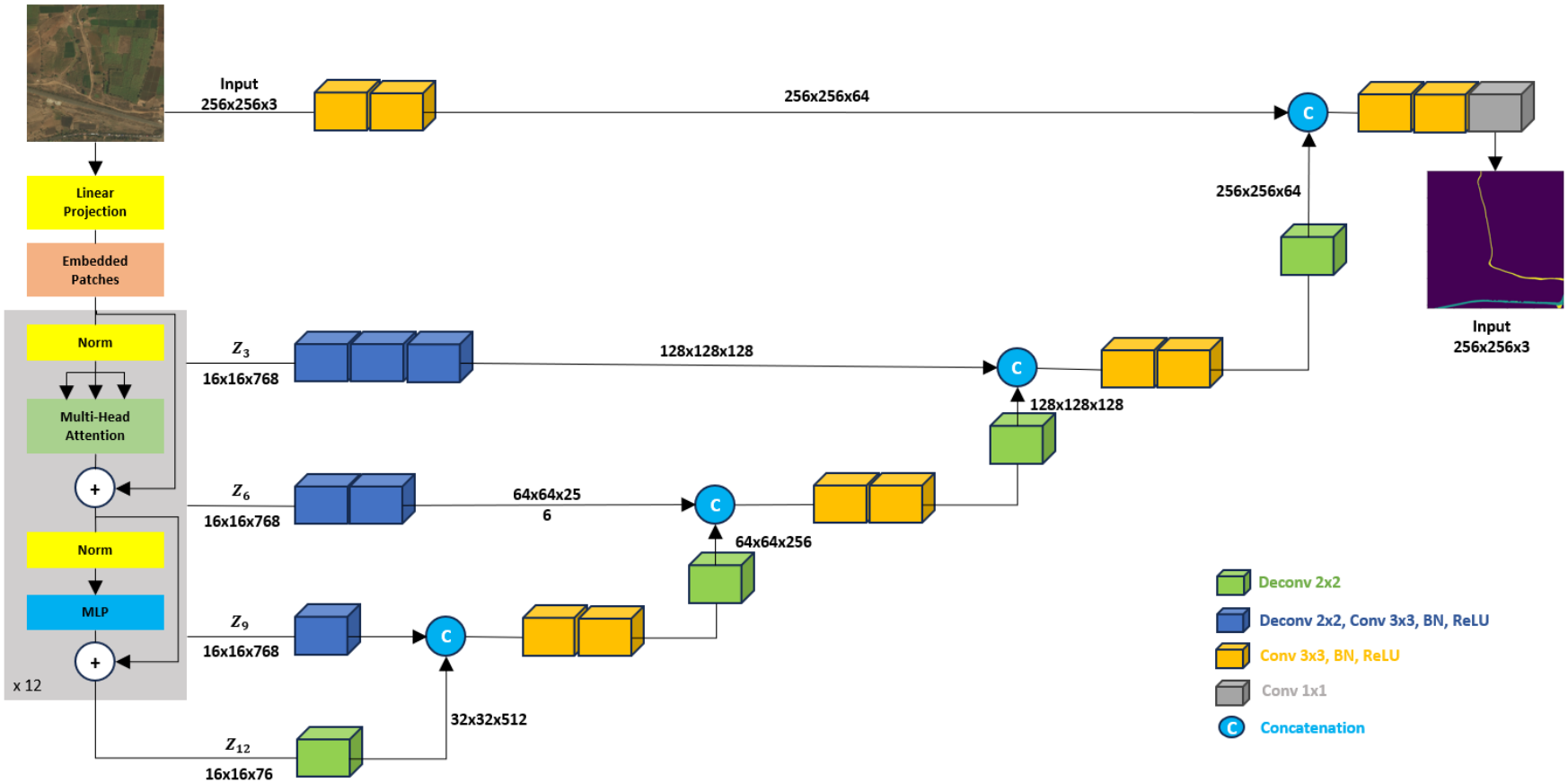


Figura 14. Arquitectura UNETR

4.3.6. Seccionamiento de imágenes en parches

Habiendo definido propiamente nuestro modelo de segmentación, es evidente que las dimensiones de las imágenes de nuestro dataset no se ajustan al modelo, por lo que las imágenes deben pasar por un procesamiento a fin de poder utilizarlas para ajustar nuestro modelo. Sabiendo que el modelo esperar imágenes de las dimensiones 256x256, lo que se procede a realizar es dividir cada imagen en parches (*patches*) del mismo tamaño, por lo que de una imagen original de 1024x1024 se obtendrían 16 imágenes de 256x256, con esto nuestro dataset, definido en la Tabla 4, se transformaría de la siguiente forma:

Tabla 5. Dataset imágenes de 256x256

Grupo	Etiqueta	Número de imágenes 256x256
Pavimentadas	paved	151 200
No pavimentadas	not paved	154 720
Both	paved y not paved	33 744
Total	**-**	**339 664**

En la Tabla 5, se puede ver el número total de imágenes que serán utilizadas para el entrenamiento del modelo.

4.3.7. Desestimación de parches

En la Figura 12, se puede apreciar un ejemplo de las etiquetas originales de nuestro dataset, al haber pasado por el proceso de la sección 4.3.6, obtendremos imágenes de 256x256 que solo contienen el fondo (o *background*) como contenido, al utilizar éstas en el ajuste del modelo contribuirían muy poco, por lo que se realiza un descarte de este tipo de imágenes considerando un umbral de información contenida en la imagen con el fin de pulir un poco más nuestro conjunto de imágenes. **Después de este proceso se obtienen 162 840 imágenes en total de dimensiones de 256x256.**

4.3.8. Distribución del dataset

Las características del dataset se han ido exponiendo a lo largo de las secciones previas, esta sección se dedica a exponer la distribución realizada para un eventual ajuste del modelo. La distribución en los grupos de entrenamiento, validación y prueba se detallan a en la Tabla 6:

Tabla 6. Distribución del dataset

Dimensiones imagen	Entrenamiento	Validación	Prueba	Total
246x256	154 799	3 970	4 071	162 840

4.3.9. Implementación de métricas adecuadas para problemas de segmentación

Para un problema de segmentación no sería idóneo considerar el *accuracy* como métrica, ya que, rápidamente se obtendría valores muy altos, los cuales no reflejan con exactitud el performance del modelo, esto debido a que las vías ocupan solo una pequeña porción de la imagen y la mayoría de esta corresponde al fondo, esto nos lleva a un desbalance en los datos (muchos más pixeles de fondo que de vías), por lo cual es más apropiado usar otras métricas que consideren este tipo de escenarios, tales como *F1 Score* y el coeficiente de similitud Jaccard, bastante considerados en problemas de segmentación.

Métricas más apropiadas que el *accuracy*, para un problema de segmentación, son las denominadas *precision* y *recall* (que toma valores entre 0 y 1, mientras más próximo a 1 se espera mejores resultados). Una forma de definir una sola métrica que incorpora aspectos de ambas métricas anteriores es la denominada *F-Measure*, definida como:

$$F_\beta = \frac{(\beta^2 + 1) * precision * recall}{\beta^2 * precision + recall}$$

Donde:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

TP, FP y FN representan los números de verdaderos positivos, falsos positivos y falsos negativos, respectivamente.

El parámetro β pondera la importancia del *precision* y *recall*. Valores para el parámetro $\beta > 1$ favorecen al *recall*, mientras que $\beta < 1$ favorece a la métrica *precision*. Cuando $\beta = 1$, ambas métricas están igualmente balanceadas; esta es la métrica más usada, la cual es llamada *F1 Score* (F_1):

$$F_1 = \frac{2 * precision * recall}{precision + recall}$$

El coeficiente de similitud Jaccard es una de las métricas ampliamente usadas en problemas de segmentación estos últimos años, este compara la similitud entre dos conjuntos de datos. La medición del coeficiente Jaccard entre dos conjuntos de datos es el resultado de la división entre el número de características que son comunes dividido sobre el número de propiedades [45], como se muestra a continuación:

$$Jaccard\ coef = \frac{|A \cap B|}{|A \cup B|}$$

La implementación de las métricas se ha llevado a cabo solamente con la librería *Tensorflow*, ésta nos ofrece interfaces para el manejo de vectores, las cuales fueron utilizadas para llevar a cabo la implementación de las fórmulas antes mencionadas.

CAPÍTULO V: EXPERIMENTACIÓN Y RESULTADOS

En la actual sección, se presenta el proceso de entrenamiento, así como los resultados obtenidos luego de esta etapa y el testeo de los modelos de segmentación propuestos. Finalmente, se ofrece una discusión de los resultados obtenidos.

5.1. Entrenamiento del modelo de segmentación

Cumpliendo los pasos definidos en el capítulo anterior, como la elaboración de nuestro dataset, la implementación de los modelos en Python e implementado las métricas a fin de evaluar el performance del modelo.

Se ha realizado numerosos entrenamientos sobre las arquitecturas presentadas, cada una con diferentes valores para los hiper parámetros. Se puede observar que el proceso es relativamente costoso, computacionalmente hablando, y que el tiempo va a depender mucho del número de imágenes de entrenamiento, numero de épocas, tasa de aprendizaje y tamaño por lote, así como también del hardware disponible.

El entrenamiento y evaluación del modelo se realizaron en el hardware descrito en la sección 0. Para la red neuronal, se obtuvo un menor valor en la función de pérdida con los siguientes valores para los hiper parámetros y número de imágenes de entrenamiento:

Tabla 7: Hiper parámetros para ajuste de modelos

Modelo / **Hiper parámetros**	**U-Net**	**UNETR**
Numero de épocas	50	50
Tamaño por lote	16	15
Tasa de aprendizaje	0.0001	0.0001
Imágenes de entrenamiento	154 799	154 799

De la Tabla 7 se puede ver que, el número de muestras utilizadas para calcular la pérdida (*loss*) y actualizar los pesos de las capas fue escogida teniendo en cuenta la memoria disponible en la tarjeta gráfica, y también que, a mayor tamaño, el entrenamiento se volvería más lento, ya que, los cálculos incrementarían. Además, que, se ha observado en la práctica que cuando se utiliza un lote más grande

se degrada la calidad del modelo, medida por su capacidad de generalización [46]. Por otro lado, según [47], existe una correlación entre la tasa de aprendizaje y el tamaño por lotes, por lo que, a un tamaño por lotes pequeño, se sugiere una taza de aprendizaje pequeña. El valor definido para el presente trabajo es de 0.0001.

La evolución del aprendizaje de los modelos, medido a través de la pérdida, se puede apreciar en la Figura 15 y Figura 16:

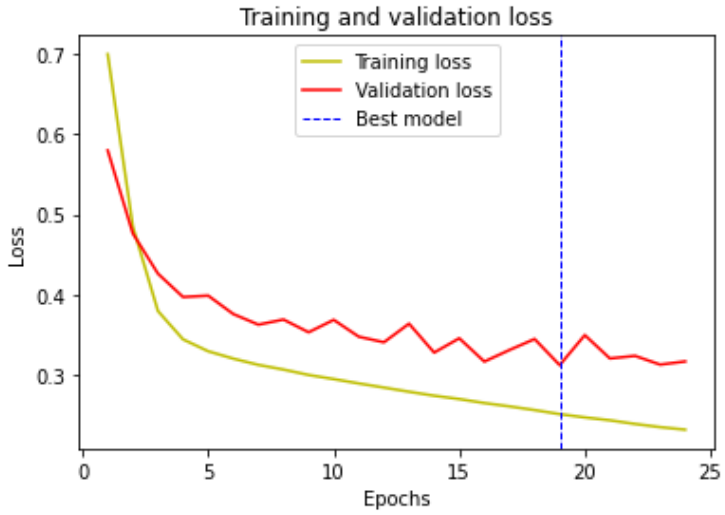


Figura 15. Pérdida (*loss*) vs Épocas (*Epochs*) – U-Net

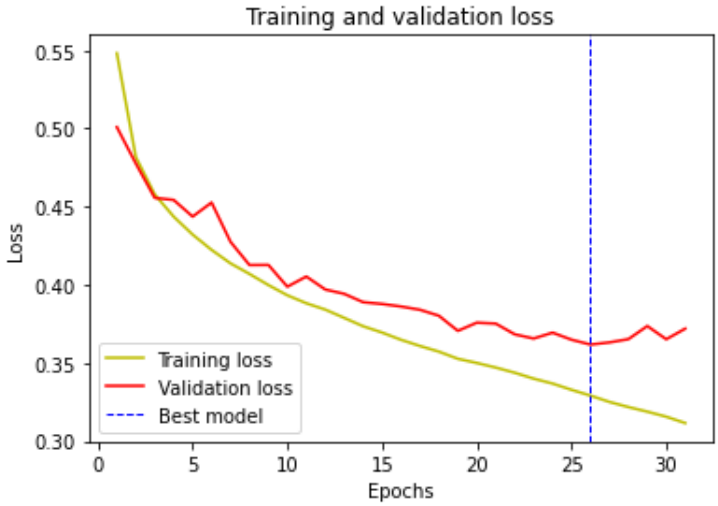


Figura 16. Pérdida (*loss*) vs Épocas (*Epochs*) – UNETR

Asimismo, la evolución del performance del modelo, medido a través de las métricas definidas anteriormente se aprecia en las siguientes figuras:

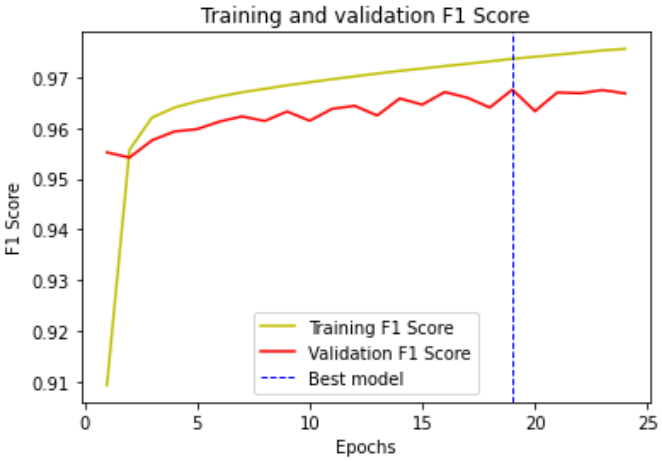


Figura 17. *F1 Score* vs Épocas (*Epochs*) – U-Net

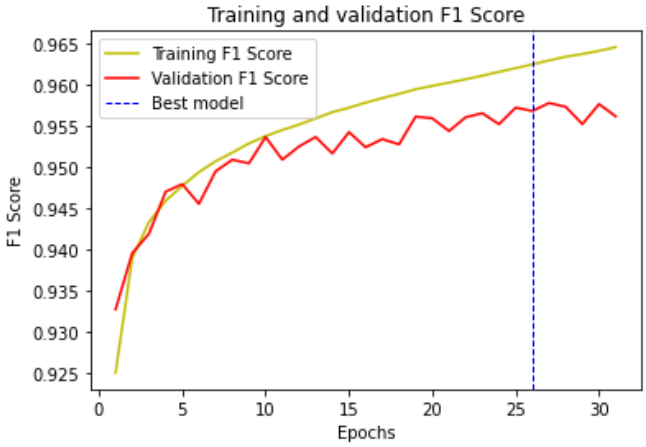


Figura 18. *F1 Score* vs Épocas (*Epochs*) – UNETR

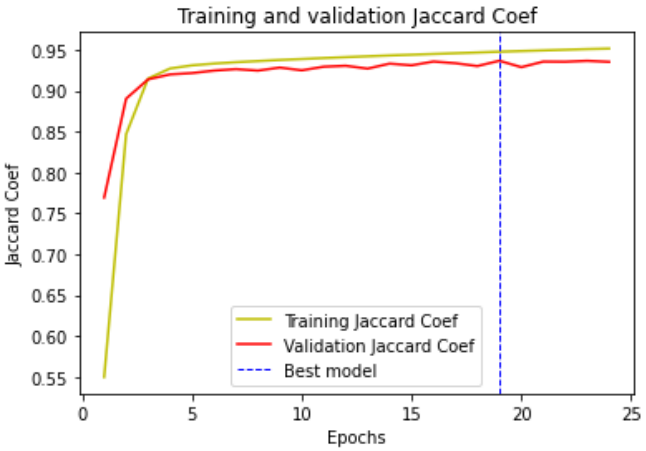


Figura 19. Coeficiente Jaccard vs Épocas (*Epochs*) – U-Net

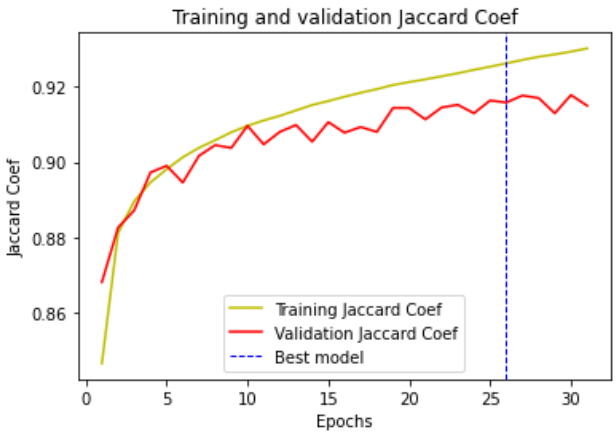


Figura 20. Coeficiente Jaccard vs Épocas (*Epochs*) – UNETR

La Figura 15 y Figura 16 nos muestra el proceso de generalización del modelo. Se puede ver que la pérdida en validación es menor que la pérdida en entrenamiento en las épocas iniciales, así como el decrecimiento de estos valores suele ser un tanto acelerado en épocas intermedias. Posteriormente, la pérdida tiende a disminuir en su aceleración de decrecimiento en las épocas finales. La curva roja, graficada en la Figura 15, presenta una suavidad llevada a lo largo de todas las épocas e incluso las finales, a diferencia de Figura 16, que en épocas finales parecería que tendiera eventualmente a un sobreajuste.

En las demás graficas se puede observar cómo aumenta la calidad de la segmentación conforme las épocas avanzas. De manera similar a la pérdida, el crecimiento es muy acelerado inicialmente, no obstante, conforme se avanza en las épocas, este incremento tiende a ser más moderado. En Figura 19, ambas curvas mantienen una relación constante en la distancia entre ellas como en su crecimiento, distinto a lo mostrado en Figura 20, que la distancia entre estas curvas es muy variada conforme avanza en épocas.

Por otro lado, en cada una de estas figuras también se distingue una línea punteada vertical, el cual hace referencia al momento en que se presenta el mejor modelo. Si bien inicialmente se ha considerado 50 épocas para el entrenamiento, no se llegaron a estas 50 épocas, ya que, el mejor modelo ocurre en la época 19 para U-Net y 26 para UNETR, y considerando una paciencia de 5 épocas, las ultimas épocas fueron 24 y 31, respectivamente. La Tabla 8, muestra las estadísticas obtenidas para los mejores modelos:

Tabla 8. Estadísticas de los mejores modelos

Métrica	U-Net		UNETR	
	Entrenamiento	**Validación**	**Entrenamiento**	**Validación**
Pérdida	0.2515	0.3125	0.3291	0.3617
F1 Score	0. 9737	0. 9676	0. 9624	0. 9568
Coeficiente Jaccard	0. 9478	0. 9367	0.9263	0. 9159
Tiempo	21hrs 00min		67hrs 10min	

Importante mencionar que el tiempo dado en la Tabla 8 es relativo. Si bien éste es un poco considerable, este tiempo podría variar dependiendo de los hiper parámetros, algo ya mencionado anteriormente, pero también depende de en donde se realice el entrenamiento. Al realizar el entrenamiento en un CPU, el tiempo sería exorbitante, pero al llevar esta labor a un GPU, como se ha realizado en esta investigación, el tiempo aminora considerablemente; así como también podría reducirse aún más si se considera más de una tarjeta gráfica.

5.2. Evaluación del modelo de segmentación

En este apartado, se procede a medir el performance de los modelos mediante la pérdida y las métricas definidas anteriormente. De igual forma se desarrolla un análisis visual de las salidas obtenidas por los modelos de segmentación.

5.2.1. Evaluación cuantitativa

Para realizar la evaluación se obtiene un conjunto de prueba, definido en la sección 4.3.8, el cual dispone de 4 071 imágenes. Con este conjunto, el modelo genera predicciones, por lo que se obtienen mapas de segmentación. La evaluación consiste en comparar la salida del modelo de segmentación con la etiqueta (*ground truth*) de cada imagen, en otras palabras, se compara la inferencia de los valores esperados.

Tabla 9. Evaluación cuantitativa de los mejores modelos

Modelo	Pérdida	*F1 Score*	Coeficiente Jaccard
U-Net	**0.3093**	**0.9691**	**0.9397**
UNETR	0.3665	0.9579	0.9180

En la Tabla 9, se puede apreciar que los resultados obtenidos en la evaluación cuantitativa guardan coherencia con los resultados en validación. Incluso, la calidad de la segmentación, medido con las métricas *F1 Score* y coeficiente Jaccard superan ligeramente a estos resultados obtenidos, presentes en la Tabla 8. Además, se distingue que los valores obtenidos en el modelo U-Net son superiores a los valores obtenidos por UNETR, para ser precisos, la diferencia obtenida entre ambos modelos en las métricas *F1 Score* y coeficiente Jaccard ascienden a 1.12% y 2.17%, respectivamente, los cuales pueden ser determinantes en un problema de segmentación.

5.2.2. Evaluación cualitativa

Adicionalmente a la evaluación cuantitativa utilizando las métricas mencionadas, se realiza un análisis visual cualitativo de los mapas de segmentación generados por el modelo. Esto permite complementar los resultados obtenidos mediante el análisis cuantitativo, así como determinar si existe relación entre estos dos tipos de análisis.

La Figura 21, compara de manera visual las predicciones del modelo de segmentación propuesto con la etiqueta de cada imagen de prueba. Se observa que, en la mayoría de los casos, las predicciones del modelo U-Net coinciden con las etiquetas, incluso este modelo es capaz de inferir correctamente aquellas vías que no fueron tomadas en cuenta por la etiqueta, independientemente si son vías pavimentadas o no, esto da a relucir un correcto aprendizaje del modelo adquirido durante el entrenamiento. Por su parte, UNETR también es poseedor de esta cualidad, como se evidencia en Figura 21(p), pero en menor frecuencia e incluso confundiendo una vía no pavimentada por una pavimentada (Figura 21(l)). Por otro lado, las predicciones realizadas por este segundo modelo no logran definir completamente aquellas vías que se infiere, quedando inconclusa en la mayoría de los casos, como por ejemplo en Figura 21(d y h), a diferencia de Figura 21(c) que si logra completitud en la mayoría de sus vías.

Asimismo, en Figura 21(g), se puede observar que el modelo U-Net logra obtener una segmentación mucho mejor definida. Comparando la etiqueta con la predicción, de igual manera la predicción logra agregar mejor definición a los bordes de las líneas verticales.

Finalmente, se ha podido apreciar que las inferencias obtenidas de los modelos de segmentación son en general bastante buenas, sobre todo del primer modelo. No obstante, se presenta intencionalmente Figura 21(s y t), donde se puede distinguir un pequeño segmento predicho como una vía no pavimentada (amarilla), el cual fue inferido correctamente pero no en su totalidad, esto indica que algunas veces para los modelos es difícil segmentar vías que están siendo obstruidas visualmente por algún objeto, en este caso arbustos.

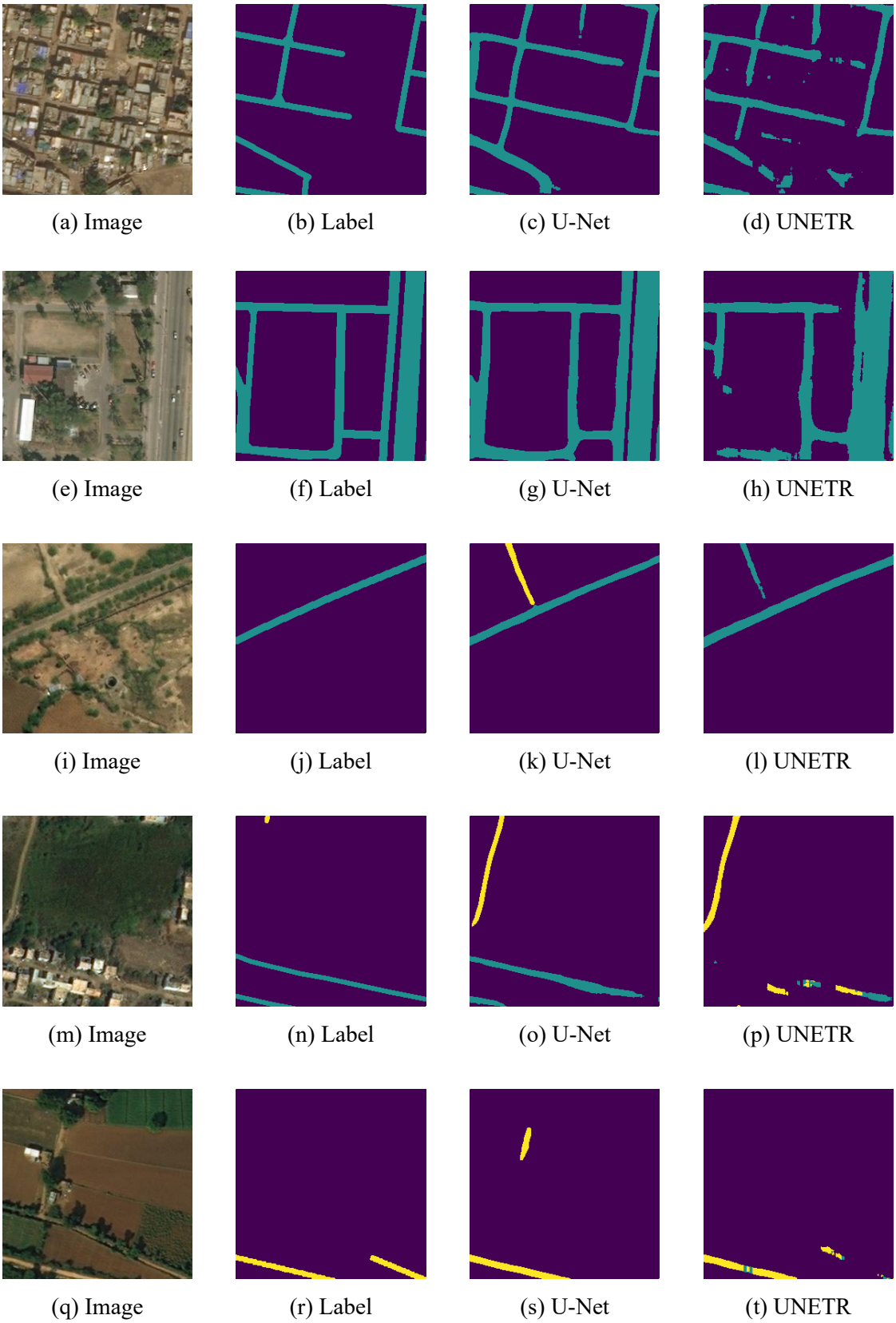


Figura 21. Imágenes de prueba: (a), (e), (i), (m) y (q). Etiquetas: (b), (f), (j), (n) y (r). Predicciones U-Net: (c), (g), (k), (o) y (s). Predicciones UNETR: (d), (h), (l), (p) y (t).

5.3. Discusión de resultados

En las secciones anteriores se ha implementado dos modelos de segmentación, los cuales fueron entrenados, así como también se ha procedido a evaluar la calidad de los mapas de segmentación producidos por la inferencia del modelo, estas evaluaciones se realizaron de manera cuantitativa y también de manera cualitativa, mediante observaciones visuales. A lo largo de este trabajo desarrollado se observaron resultados interesantes los cuales se discute a continuación:

- La arquitectura propuesta es computacionalmente menos costosa, comparada con otros enforques presentes en el estado del arte. El tiempo en entrenamiento va a depender también de los hiper parámetros seleccionados. Sobre esto, es importante mencionar que el filtro (desestimación de parches) realizado durante la construcción de nuestro dataset aporta significativamente a reducir el costo en tiempo en el entrenamiento, ya que, no se considera aquellas imágenes que no superen el umbral de información contenida en ellas.
- En la evaluación cualitativa, se ha podido presenciar que el performance de ambos modelos son aceptables, y que ambos podría ser una herramienta considerable para la segmentación multiclase de vías, sin embargo, es importante recalcar que, los resultados de U-Net superan notoriamente a los de UNETR en el análisis visual realizado. Aunque para los modelos algunas veces es difícil inferir la vía que esta siento obstruida visualmente por algún objeto, los modelos son capaces de inferir no solo lo esperado en la etiqueta, sino que también detecta aquellas vías que no fueron consideradas por ella. Asimismo, estos son capaz de mejorar la calidad de los bordes de las vías, aplicando mucho más detalle que lo esperado en las etiquetas. Todo esto deja en evidencia la buena capacidad de generalización de los modelos de segmentación semántica.
- De igual forma a la evaluación cualitativa, en la evaluación cuantitativa, UNETR es superado por U-Net en las métricas evaluadas. La diferencia en el performance evaluado asciende en un 1.12% y 2.17% sobre las métricas *F1 Score* y coeficiente Jaccard, respectivamente, si bien la diferencia podría parecer no muy distante, esto podría hacer una diferencia notable en problemas de visión por computador, lo cual se puede corroborar con la evaluación cualitativa, donde se ha obtenido mejores resultados en U-Net, por lo que se encuentra a este modelo mejor adaptado para este problema en específico. Como se dijo anteriormente, no existen trabajos que aborden el problema de manera multiclase, por lo cual, una comparación con los trabajos presentes en la literatura no es posible, sin embargo, los resultados obtenidos en el presente trabajo servirán como marco de referencia para posteriores investigaciones.

CAPÍTULO VI: CONCLUSIONES

Finalmente, después de haber realizado la implementación del modelo de segmentación semántica, en este capítulo, se indican las principales conclusiones logradas en el presente trabajo, así como también se establece posibles trabajos a futuro que pueden abordarse partiendo del modelo propuesto.

6.1. Conclusiones

En relación al objetivo general, a los objetivos específicos y a los resultados obtenidos se puede concluir que:

- Se ha logrado desarrollar dos modelos, basado en redes convolucionales, para la segmentación multiclase automática de vías en imágenes satelitales. Las fases de elaboración, construcción, entrenamiento y evaluación del los modelos propuestos en la metodología han sido ideadas cuidadosamente y en orden, a fin de lograr adecuados modelos de segmentación, el mismo que ha obtenido resultados muy favorables en las respectivas evaluaciones.
- Se definió un procedimiento para preprocesar y construir una base de datos de imágenes, que sirvan para cada una de las fases que atraviesa un modelo neuronal, llámese entrenamiento, validación y prueba. Se espera que este procedimiento sirva de guía para trabajos en donde no se dispone de conjunto de datos para realizar investigación.
- Se ha elegido las métricas *F1 Score* y coeficiente de similitud Jaccard como idóneas para medir la calidad de segmentación, esto debido a que responde de manera apropiada a problemas de desbalanceo de datos, problema que es bastante común en segmentación de imágenes.
- Se ha implementado dos modelos convolucionales para la segmentación multiclase de vías en imágenes satelitales. La arquitectura que presenta mejor rendimiento para este problema en específico es la denominada U-Net.
- Se ha evaluado el desempeño de los modelos de manera cualitativa y cuantitativa. La evaluación cualitativa se realizó mediante comparaciones visuales entre la etiqueta original de la imagen y la inferencia del modelo. En estas pruebas, los modelos fueron capaces de segmentar las vías esperadas (marcadas en las etiquetas) pero también aquellas

que no fueron consideradas por la etiqueta, lo cual da a relucir una adecuada generalización de los modelos adquirido en el periodo de entrenamiento. Por otro lado, el rendimiento del mejor modelo medido numéricamente asciende en al menos 93.97% en todas las métricas consideradas, el cual es un valor bastante alentador y coincide con lo obtenido en la evaluación cualitativa.

- El presente trabajo corresponde a uno de los primeros en abordar el problema de segmentación de vías de manera multiclase, por lo que se espera que sea de utilidad para futuros trabajos relacionados.

6.2. Trabajo a futuro

A partir del trabajo desarrollado y considerando ideas de mejora pensadas en el camino se puede sugerir los siguientes trabajos a futuro:

- Construir un dataset variado con imágenes satelitales que no solo pertenezcan a datasets abiertos (que generalmente podrían ser de otras partes del mundo), sino también disponer de imágenes de zonas puntuales de interés, como el interior del País. E incluso también poder considerar un mayor número de etiquetas (dentro del marco vial), de esta forma se incrementaría la complejidad para un eventual modelo. Esto con el fin de hacer el dataset más rico y diverso, ya que, el Perú dispone de una infraestructura vial bastante heterogénea.
- Entrenar el modelo U-Net con un número mayor de imágenes, y que este nuevo dataset contengan imágenes ricas en diversas zonas del Perú. Al reentrenar el modelo con imágenes con nuevas características, se esperaría mejores resultados a nivel cuantitativo y cualitativo.
- Implementar un nuevo modelo o algoritmo, que tome como entrada el mapa de segmentación generado por el modelo propuesto, a fin de que proporcione información adicional, por ejemplo, el área o distancia de la vía contenida en la imagen satelital.

REFERENCIAS

[1] Gobierno del Perú, «Plataforma digital única del Estado Peruano,» 21 Diciembre 2022. [En línea]. Available: https://www.gob.pe/institucion/mtc/informes-publicaciones/344790-estadistica-infraestructura-de-transportes-infraestructura-vial.

[2] J. Long, E. Shelhamer y T. Darrell, «Fully Convolutional Networks for Semantic Segmentation,» 2015.

[3] H. Noh, S. Hong y B. Han, «Learning Deconvolution Network for Semantic Segmentation,» 2015.

[4] O. Ronneberger, P. Fischer y T. Brox, «U-NET: Convolutional Networks for Biomedical Image Segmentation,» 2015.

[5] S. R. Yoshida, Computer Vision, New York: Nova Science Publishers, Inc., 2011.

[6] R. Szeliski, Computer Vision: Algorithms and Applications, New York: Springer, 2011.

[7] R. Klette, Concise Computer Vision : An Introduccion into Theory and Algorithms, New York: Springer, 2014.

[8] A. S. Chauhan, S. Silakari y M. Dixit, «Image Segmentation Methods: A survey Approach,» 2014.

[9] S. Jayaraman, S. Esakkirajan y S. Veerakumar, Digital Image Processing, New Delhi: McGraw Hill Educatgion, 2019.

[10] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz y D. Terzopoulos, «Image Segmentation Using Deep Learning: A Survey,» 2020.

[11] M. Thoma, «A Survey of Semantic Segmentation,» 2016.

[12] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick y P. Dollar´, «Microsoft COCO: Common Objects in Context,» 2015.

[13] IEEE Board of Directors, «Artificial Intelligence,» 24 Junio 2019. [En línea]. Available: https://globalpolicy.ieee.org/wp-content/uploads/2019/06/IEEE18029.pdf. [Último acceso: 26 Octubre 2023].

[14] S. A. Seshia, D. Sadigh y S. S. Sastry, «Communications of the ACM,» ACM, Julio 2022. [En línea]. Available: https://cacm.acm.org/magazines/2022/7/262079-toward-verified-artificial-intelligence/fulltext#:~:text=Artificial%20intelligence%20(AI)%20is%20a,example%2C%20see%20Russell%20and%20Norvig.. [Último acceso: 26 Octubre 2023].

[15] IBM, «IBM Cloud Paks,» IBM, 14 Julio 2022. [En línea]. Available: https://www.ibm.com/docs/en/cloud-paks/1.0?topic=cloudpaks_start/ibm-process-mining/user-manuals/ai_ml_platformintegration/introduction.htm. [Último acceso: 26 Octubre 2023].

[16] ACM, «ACM SELECTS,» ACM, 13 Octubre 2020. [En línea]. Available: https://selects.acm.org/selections/understanding-machine-learning. [Último acceso: 26 Octubre 2023].

[17] K. P. Murphy, Machine Learning - A Probailistic Perspective, London: The MIT Press, 2012.

[18] Y. LeCun, Y. Bengio y G. Hinton, «Deep Learning,» *Nature,* 2015.

[19] F. Q. Lauzon, «An introduction to deep learning,» *IEEE,* 2012.

[20] P. Ponce Cruz, Inteligencia Artificial con Aplicaciones a la Ingeniería, DF, México: Alfaomega, 2010.

[21] S. J. Russell y P. Norvig, Inteligencia Artificial. Un enforque moderno., Madrid: Pearson Educación S.A., 2008.

[22] I. Goodfellow, Y. Bengio y A. Courville, Deep Learning, NY: The MIT Press, 2016.

[23] S. K. Vasudevan, S. R. Pulari y S. Vasudevan, Deep Learning : A comprehensive Guide, Abingdon: CRC Press, 2022.

[24] D. Jurafsky y J. H. Martin, Speech and Language Processing, 2022.

[25] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit y N. Houlsby, «AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE,» *ICLR,* 2021.

[26] Ministerio de Transportes y Comunicaciones, «Proyecto Reglamento Nacional de Gestion de la Infraestructura Vial,» El Peruano, Lima, 2006.

[27] Agencia Nacional de Seguridad Vial, «Infraestructura Vial: Factor de Riesgo de la Seguridad Vial,» Gobierno de Argentina, [En línea]. Available: https://www.argentina.gob.ar/seguridadvial/observatoriovialnacional/infraestructura-vial-factor-de-riesgo-de-la-seguridad-vial. [Último acceso: 12 Diciembre 2023].

[28] D. Guo, X. Li y W. Lou, «Detection and analysis of road information based on optical satellite image,» *IEEE,* 2021.

[29] E. Kurbatova y V. Laylina, «Detection of Roads from Images Based on Edge Segmentation and Morphological Operations,» *IEEE,* 2019.

[30] N. S. Kumar, B. Sukanya, B. Mohan y G. Prathibha, «Extraction of Roads from Satellite Images Based on Edge Detection,» 2017.

[31] A. Maarir y B. Bouikhalene, «Roads Detection from Satellite Images Based on Active Contour Model and Distance Transform,» *IEEE,* 2016.

[32] M. Maboudi, J. Amini, M. Hahn y M. Saati, «Road Network Extraction from VHR Satellite Images Using Context Aware Object Feature Integration and Tensor Voting,» *Remote Sensing,* 2016.

[33] B. Sirmacek y C. Unsalan, «Road detection from remotely sensed images using color features,» *IEEE,* 2011.

[34] T. Li, M. Comer y J. Zerubia, «Feature Extraction and Tracking of CNN Segmentations for Improved Road Detection from Satellite Imagery,» *IEEE,* 2019.

[35] V. Yerram, H. Takeshita, Y. Iwahori, Y. Hayashi, M. K. Bhuyan, S. Fukui, B. Kijsirikul y A. Wang, «Extraction and Calculation of Roadway Area from Satellite Images Using Improved Deep Learning Model and Post-Processing,» *Journal of Imaging,* 2022.

[36] H. Ghandorh, W. Boulila, S. Masood, A. Koubaa, F. Ahmed y J. Ahmad, «Semantic Segmentation and Edge Detection—Approach to Road Detection in Very High Resolution Satellite Images,» *Remote Sensing,* 2022.

[37] T. K. Behera, P. K. Sa, M. Nappi y S. Bakshi, «Satellite IoT Based Road Extraction from VHR Images Through Superpixel-CNN Architecture,» *Big Data Research,* 2022.

[38] G. Cheng, C. Wu, Q. Huang, Y. Meng, J. Shi, J. Chen y D. Yan, «Recognizing road from satellite images by structured neural network,» *Neurocomputing,* 2019.

[39] H. R. R. Bakhtiari, A. Abdollahi y H. Rezaeian, «Semi automatic road extraction from digital images,» *The Egyptian Journal of Remote Sensing and Space Science,* 2017.

[40] A. Ramirez, A. Pacheco y J. Telles, «Mapping vegetation, water bodies and urban areas in PeruSAT-1 satellite imagery,» *IEEE,* 2019.

[41] J. Gonzalez, K. Sankaran, V. Ayma y C. Beltran, «APPLICATION OF SEMANTIC SEGMENTATION WITH FEW LABELS IN THE DETECTION OF WATER BODIES FROM PERUSAT-1 SATELLITE'S IMAGES,» *IEEE,* 2020.

[42] W. Yin, M. Qian, L. Wang, J. Qi y H. Lu, «Road extraction from satellite images with iterative cross-task feature enhancement,» *Neurocomputing,* 2022.

[43] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia y R. Raskar, «DeepGlobe 2018: A Challenge to Parse the Earth through Satellite Images,» 2018.

[44] A. Hatamizadeh, Y. Tang, V. Nath, D. Yang, A. Myronenko, B. Landman, H. R. Roth y D. Xu, «UNETR: Transformers for 3D Medical Image Segmentation,» 2021.

[45] S. Niwattanakul, J. Singthongchai, E. Naenudorn y S. Wanapu, «Using of Jaccard Coefficient for Keywords Similarity,» *Proceedings of the International MultiConference of Engineers and Computer Scientists 2013 Vol I,* 2013.

[46] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy y P. Tang, «ON LARGE-BATCH TRAINING FOR DEEP LEARNING: GENERALIZATION GAP AND SHARP MINIMA,» *ICLR,* 2017.

[47] I. Kandel y M. Castelli, «The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset,» *KICS,* 2020.

ANEXOS

Anexo A: Repositorio de código

El presente anexo contiene el enlace del repositorio donde se aloja el código implementado en el presente trabajo de investigación: Código de construcción de muestras y modelos

Anexo B: Arquitectura de los modelos convolucionales

Modelo U-Net

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 256, 256, 3)]	0	[]
conv2d (Conv2D)	(None, 256, 256, 64)	1792	['input_1[0][0]']
batch_normalization (BatchNormalization)	(None, 256, 256, 64)	256	['conv2d[0][0]']
activation (Activation)	(None, 256, 256, 64)	0	['batch_normalization[0][0]']
conv2d_1 (Conv2D)	(None, 256, 256, 64)	36928	['activation[0][0]']
batch_normalization_1 (BatchNormalization)	(None, 256, 256, 64)	256	['conv2d_1[0][0]']
activation_1 (Activation)	(None, 256, 256, 64)	0	['batch_normalization_1[0][0]']
dropout (Dropout)	(None, 256, 256, 64)	0	['activation_1[0][0]']
max_pooling2d (MaxPooling2D)	(None, 128, 128, 64)	0	['dropout[0][0]']
conv2d_2 (Conv2D)	(None, 128, 128, 128)	73856	['max_pooling2d[0][0]']
batch_normalization_2 (BatchNormalization)	(None, 128, 128, 128)	512	['conv2d_2[0][0]']
activation_2 (Activation)	(None, 128, 128, 128)	0	['batch_normalization_2[0][0]']
conv2d_3 (Conv2D)	(None, 128, 128, 128)	147584	['activation_2[0][0]']
batch_normalization_3 (BatchNormalization)	(None, 128, 128, 128)	512	['conv2d_3[0][0]']
activation_3 (Activation)	(None, 128, 128, 128)	0	['batch_normalization_3[0][0]']
dropout_1 (Dropout)	(None, 128, 128, 128)	0	['activation_3[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 128)	0	['dropout_1[0][0]']
conv2d_4 (Conv2D)	(None, 64, 64, 256)	295168	['max_pooling2d_1[0][0]']
batch_normalization_4 (BatchNormalization)	(None, 64, 64, 256)	1024	['conv2d_4[0][0]']
activation_4 (Activation)	(None, 64, 64, 256)	0	['batch_normalization_4[0][0]']
conv2d_5 (Conv2D)	(None, 64, 64, 256)	590080	['activation_4[0][0]']
batch_normalization_5 (BatchNormalization)	(None, 64, 64, 256)	1024	['conv2d_5[0][0]']
activation_5 (Activation)	(None, 64, 64, 256)	0	['batch_normalization_5[0][0]']
dropout_2 (Dropout)	(None, 64, 64, 256)	0	['activation_5[0][0]']
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 256)	0	['dropout_2[0][0]']
conv2d_6 (Conv2D)	(None, 32, 32, 512)	1180160	['max_pooling2d_2[0][0]']
batch_normalization_6 (BatchNormalization)	(None, 32, 32, 512)	2048	['conv2d_6[0][0]']
activation_6 (Activation)	(None, 32, 32, 512)	0	['batch_normalization_6[0][0]']
conv2d_7 (Conv2D)	(None, 32, 32, 512)	2359808	['activation_6[0][0]']
batch_normalization_7 (BatchNormalization)	(None, 32, 32, 512)	2048	['conv2d_7[0][0]']
activation_7 (Activation)	(None, 32, 32, 512)	0	['batch_normalization_7[0][0]']
dropout_3 (Dropout)	(None, 32, 32, 512)	0	['activation_7[0][0]']
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 512)	0	['dropout_3[0][0]']
conv2d_8 (Conv2D)	(None, 16, 16, 1024)	4719616	['max_pooling2d_3[0][0]']

batch_normalization_8 (BatchNormalization)	(None, 16, 16, 1024)	4096	['conv2d_8[0][0]']
activation_8 (Activation)	(None, 16, 16, 1024)	0	['batch_normalization_8[0][0]']
conv2d_9 (Conv2D)	(None, 16, 16, 1024)	9438208	['activation_8[0][0]']
batch_normalization_9 (BatchNormalization)	(None, 16, 16, 1024)	4096	['conv2d_9[0][0]']
activation_9 (Activation)	(None, 16, 16, 1024)	0	['batch_normalization_9[0][0]']
dropout_4 (Dropout)	(None, 16, 16, 1024)	0	['activation_9[0][0]']
up_sampling2d (UpSampling2D)	(None, 32, 32, 1024)	0	['dropout_4[0][0]']
concatenate (Concatenate)	(None, 32, 32, 1536)	0	['up_sampling2d[0][0]', 'dropout_3[0][0]']
conv2d_10 (Conv2D)	(None, 32, 32, 512)	7078400	['concatenate[0][0]']
batch_normalization_10 (BatchNormalization)	(None, 32, 32, 512)	2048	['conv2d_10[0][0]']
activation_10 (Activation)	(None, 32, 32, 512)	0	['batch_normalization_10[0][0]']
conv2d_11 (Conv2D)	(None, 32, 32, 512)	2359808	['activation_10[0][0]']
batch_normalization_11 (BatchNormalization)	(None, 32, 32, 512)	2048	['conv2d_11[0][0]']
activation_11 (Activation)	(None, 32, 32, 512)	0	['batch_normalization_11[0][0]']
dropout_5 (Dropout)	(None, 32, 32, 512)	0	['activation_11[0][0]']
up_sampling2d_1 (UpSampling2D)	(None, 64, 64, 512)	0	['dropout_5[0][0]']
concatenate_1 (Concatenate)	(None, 64, 64, 768)	0	['up_sampling2d_1[0][0]', 'dropout_2[0][0]']
conv2d_12 (Conv2D)	(None, 64, 64, 256)	1769728	['concatenate_1[0][0]']
batch_normalization_12 (BatchNormalization)	(None, 64, 64, 256)	1024	['conv2d_12[0][0]']
activation_12 (Activation)	(None, 64, 64, 256)	0	['batch_normalization_12[0][0]']
conv2d_13 (Conv2D)	(None, 64, 64, 256)	590080	['activation_12[0][0]']
batch_normalization_13 (BatchNormalization)	(None, 64, 64, 256)	1024	['conv2d_13[0][0]']
activation_13 (Activation)	(None, 64, 64, 256)	0	['batch_normalization_13[0][0]']
dropout_6 (Dropout)	(None, 64, 64, 256)	0	['activation_13[0][0]']
up_sampling2d_2 (UpSampling2D)	(None, 128, 128, 256)	0	['dropout_6[0][0]']
concatenate_2 (Concatenate)	(None, 128, 128, 384)	0	['up_sampling2d_2[0][0]', 'dropout_1[0][0]']
conv2d_14 (Conv2D)	(None, 128, 128, 128)	442496	['concatenate_2[0][0]']
batch_normalization_14 (BatchNormalization)	(None, 128, 128, 128)	512	['conv2d_14[0][0]']
activation_14 (Activation)	(None, 128, 128, 128)	0	['batch_normalization_14[0][0]']
conv2d_15 (Conv2D)	(None, 128, 128, 128)	147584	['activation_14[0][0]']
batch_normalization_15 (BatchNormalization)	(None, 128, 128, 128)	512	['conv2d_15[0][0]']
activation_15 (Activation)	(None, 128, 128, 128)	0	['batch_normalization_15[0][0]']
dropout_7 (Dropout)	(None, 128, 128, 128)	0	['activation_15[0][0]']
up_sampling2d_3 (UpSampling2D)	(None, 256, 256, 128)	0	['dropout_7[0][0]']
concatenate_3 (Concatenate)	(None, 256, 256, 192)	0	['up_sampling2d_3[0][0]', 'dropout[0][0]']
conv2d_16 (Conv2D)	(None, 256, 256, 64)	110656	['concatenate_3[0][0]']
batch_normalization_16 (BatchNormalization)	(None, 256, 256, 64)	256	['conv2d_16[0][0]']
activation_16 (Activation)	(None, 256, 256, 64)	0	['batch_normalization_16[0][0]']
conv2d_17 (Conv2D)	(None, 256, 256, 64)	36928	['activation_16[0][0]']
batch_normalization_17 (BatchNormalization)	(None, 256, 256, 64)	256	['conv2d_17[0][0]']
activation_17 (Activation)	(None, 256, 256, 64)	0	['batch_normalization_17[0][0]']
dropout_8 (Dropout)	(None, 256, 256, 64)	0	['activation_17[0][0]']
conv2d_18 (Conv2D)	(None, 256, 256, 3)	195	['dropout_8[0][0]']
batch_normalization_18 (BatchNormalization)	(None, 256, 256, 3)	12	['conv2d_18[0][0]']

```
 activation_18 (Activation)                  (None, 256, 256, 3)             0                ['batch_normalization_18[0][0]']
==================================================================================================================================================
Total params: 31,402,639
Trainable params: 31,390,857
Non-trainable params: 11,782
__________________________________________________________________________________________________________________________________________________
```

Modelo UNETR

```
__________________________________________________________________________________________________________________________________________________
 Layer (type)                                Output Shape                  Param #           Connected to
==================================================================================================================================================
 input_5 (InputLayer)                        [(None, 256, 768)]            0                 []
 dense_50 (Dense)                            (None, 256, 768)              590592            ['input_5[0][0]']
 tf.__operators__.add_2 (TFOpLambda)         (None, 256, 768)              0                 ['dense_50[0][0]']
 layer_normalization_48 (LayerNormalization)  (None, 256, 768)             1536              ['tf.__operators__.add_2[0][0]']
 multi_head_attention_24 (MultiHeadAttention)  (None, 256, 768)            28339968          ['layer_normalization_48[0][0]',
                                                                                             'layer_normalization_48[0][0]']
 add_48 (Add)                                (None, 256, 768)              0                 ['multi_head_attention_24[0][0]',
                                                                                             'tf.__operators__.add_2[0][0]']
 layer_normalization_49 (LayerNormalization)  (None, 256, 768)             1536              ['add_48[0][0]']
 dense_51 (Dense)                            (None, 256, 3072)             2362368           ['layer_normalization_49[0][0]']
 dropout_66 (Dropout)                        (None, 256, 3072)             0                 ['dense_51[0][0]']
 dense_52 (Dense)                            (None, 256, 768)              2360064           ['dropout_66[0][0]']
 dropout_67 (Dropout)                        (None, 256, 768)              0                 ['dense_52[0][0]']
 add_49 (Add)                                (None, 256, 768)              0                 ['dropout_67[0][0]',
                                                                                             'add_48[0][0]']
 layer_normalization_50 (LayerNormalization)  (None, 256, 768)             1536              ['add_49[0][0]']
 multi_head_attention_25 (MultiHeadAttention)  (None, 256, 768)            28339968          ['layer_normalization_50[0][0]',
                                                                                             'layer_normalization_50[0][0]']
 add_50 (Add)                                (None, 256, 768)              0                 ['multi_head_attention_25[0][0]',
                                                                                             'add_49[0][0]']
 layer_normalization_51 (LayerNormalization)  (None, 256, 768)             1536              ['add_50[0][0]']
 dense_53 (Dense)                            (None, 256, 3072)             2362368           ['layer_normalization_51[0][0]']
 dropout_68 (Dropout)                        (None, 256, 3072)             0                 ['dense_53[0][0]']
 dense_54 (Dense)                            (None, 256, 768)              2360064           ['dropout_68[0][0]']
 dropout_69 (Dropout)                        (None, 256, 768)              0                 ['dense_54[0][0]']
 add_51 (Add)                                (None, 256, 768)              0                 ['dropout_69[0][0]',
                                                                                             'add_50[0][0]']
 layer_normalization_52 (LayerNormalization)  (None, 256, 768)             1536              ['add_51[0][0]']
 multi_head_attention_26 (MultiHeadAttention)  (None, 256, 768)            28339968          ['layer_normalization_52[0][0]',
                                                                                             'layer_normalization_52[0][0]']
 add_52 (Add)                                (None, 256, 768)              0                 ['multi_head_attention_26[0][0]',
                                                                                             'add_51[0][0]']
 layer_normalization_53 (LayerNormalization)  (None, 256, 768)             1536              ['add_52[0][0]']
 dense_55 (Dense)                            (None, 256, 3072)             2362368           ['layer_normalization_53[0][0]']
 dropout_70 (Dropout)                        (None, 256, 3072)             0                 ['dense_55[0][0]']
 dense_56 (Dense)                            (None, 256, 768)              2360064           ['dropout_70[0][0]']
 dropout_71 (Dropout)                        (None, 256, 768)              0                 ['dense_56[0][0]']
```

add_53 (Add)	(None, 256, 768)	0	['dropout_71[0][0]', 'add_52[0][0]']
layer_normalization_54 (LayerNormalization)	(None, 256, 768)	1536	['add_53[0][0]']
multi_head_attention_27 (MultiHeadAttention)	(None, 256, 768)	28339968	['layer_normalization_54[0][0]', 'layer_normalization_54[0][0]']
add_54 (Add)	(None, 256, 768)	0	['multi_head_attention_27[0][0]', 'add_53[0][0]']
layer_normalization_55 (LayerNormalization)	(None, 256, 768)	1536	['add_54[0][0]']
dense_57 (Dense)	(None, 256, 3072)	2362368	['layer_normalization_55[0][0]']
dropout_72 (Dropout)	(None, 256, 3072)	0	['dense_57[0][0]']
dense_58 (Dense)	(None, 256, 768)	2360064	['dropout_72[0][0]']
dropout_73 (Dropout)	(None, 256, 768)	0	['dense_58[0][0]']
add_55 (Add)	(None, 256, 768)	0	['dropout_73[0][0]', 'add_54[0][0]']
layer_normalization_56 (LayerNormalization)	(None, 256, 768)	1536	['add_55[0][0]']
multi_head_attention_28 (MultiHeadAttention)	(None, 256, 768)	28339968	['layer_normalization_56[0][0]', 'layer_normalization_56[0][0]']
add_56 (Add)	(None, 256, 768)	0	['multi_head_attention_28[0][0]', 'add_55[0][0]']
layer_normalization_57 (LayerNormalization)	(None, 256, 768)	1536	['add_56[0][0]']
dense_59 (Dense)	(None, 256, 3072)	2362368	['layer_normalization_57[0][0]']
dropout_74 (Dropout)	(None, 256, 3072)	0	['dense_59[0][0]']
dense_60 (Dense)	(None, 256, 768)	2360064	['dropout_74[0][0]']
dropout_75 (Dropout)	(None, 256, 768)	0	['dense_60[0][0]']
add_57 (Add)	(None, 256, 768)	0	['dropout_75[0][0]', 'add_56[0][0]']
layer_normalization_58 (LayerNormalization)	(None, 256, 768)	1536	['add_57[0][0]']
multi_head_attention_29 (MultiHeadAttention)	(None, 256, 768)	28339968	['layer_normalization_58[0][0]', 'layer_normalization_58[0][0]']
add_58 (Add)	(None, 256, 768)	0	['multi_head_attention_29[0][0]', 'add_57[0][0]']
layer_normalization_59 (LayerNormalization)	(None, 256, 768)	1536	['add_58[0][0]']
dense_61 (Dense)	(None, 256, 3072)	2362368	['layer_normalization_59[0][0]']
dropout_76 (Dropout)	(None, 256, 3072)	0	['dense_61[0][0]']
dense_62 (Dense)	(None, 256, 768)	2360064	['dropout_76[0][0]']
dropout_77 (Dropout)	(None, 256, 768)	0	['dense_62[0][0]']
add_59 (Add)	(None, 256, 768)	0	['dropout_77[0][0]', 'add_58[0][0]']
layer_normalization_60 (LayerNormalization)	(None, 256, 768)	1536	['add_59[0][0]']
multi_head_attention_30 (MultiHeadAttention)	(None, 256, 768)	28339968	['layer_normalization_60[0][0]', 'layer_normalization_60[0][0]']
add_60 (Add)	(None, 256, 768)	0	['multi_head_attention_30[0][0]', 'add_59[0][0]']
layer_normalization_61 (LayerNormalization)	(None, 256, 768)	1536	['add_60[0][0]']
dense_63 (Dense)	(None, 256, 3072)	2362368	['layer_normalization_61[0][0]']
dropout_78 (Dropout)	(None, 256, 3072)	0	['dense_63[0][0]']
dense_64 (Dense)	(None, 256, 768)	2360064	['dropout_78[0][0]']
dropout_79 (Dropout)	(None, 256, 768)	0	['dense_64[0][0]']

add_61 (Add)	(None, 256, 768)	0	['dropout_79[0][0]', 'add_60[0][0]']
layer_normalization_62 (LayerNormalization)	(None, 256, 768)	1536	['add_61[0][0]']
multi_head_attention_31 (MultiHeadAttention)	(None, 256, 768)	28339968	['layer_normalization_62[0][0]', 'layer_normalization_62[0][0]']
add_62 (Add)	(None, 256, 768)	0	['multi_head_attention_31[0][0]', 'add_61[0][0]']
layer_normalization_63 (LayerNormalization)	(None, 256, 768)	1536	['add_62[0][0]']
dense_65 (Dense)	(None, 256, 3072)	2362368	['layer_normalization_63[0][0]']
dropout_80 (Dropout)	(None, 256, 3072)	0	['dense_65[0][0]']
dense_66 (Dense)	(None, 256, 768)	2360064	['dropout_80[0][0]']
dropout_81 (Dropout)	(None, 256, 768)	0	['dense_66[0][0]']
add_63 (Add)	(None, 256, 768)	0	['dropout_81[0][0]', 'add_62[0][0]']
layer_normalization_64 (LayerNormalization)	(None, 256, 768)	1536	['add_63[0][0]']
multi_head_attention_32 (MultiHeadAttention)	(None, 256, 768)	28339968	['layer_normalization_64[0][0]', 'layer_normalization_64[0][0]']
add_64 (Add)	(None, 256, 768)	0	['multi_head_attention_32[0][0]', 'add_63[0][0]']
layer_normalization_65 (LayerNormalization)	(None, 256, 768)	1536	['add_64[0][0]']
dense_67 (Dense)	(None, 256, 3072)	2362368	['layer_normalization_65[0][0]']
dropout_82 (Dropout)	(None, 256, 3072)	0	['dense_67[0][0]']
dense_68 (Dense)	(None, 256, 768)	2360064	['dropout_82[0][0]']
dropout_83 (Dropout)	(None, 256, 768)	0	['dense_68[0][0]']
add_65 (Add)	(None, 256, 768)	0	['dropout_83[0][0]', 'add_64[0][0]']
layer_normalization_66 (LayerNormalization)	(None, 256, 768)	1536	['add_65[0][0]']
multi_head_attention_33 (MultiHeadAttention)	(None, 256, 768)	28339968	['layer_normalization_66[0][0]', 'layer_normalization_66[0][0]']
add_66 (Add)	(None, 256, 768)	0	['multi_head_attention_33[0][0]', 'add_65[0][0]']
layer_normalization_67 (LayerNormalization)	(None, 256, 768)	1536	['add_66[0][0]']
dense_69 (Dense)	(None, 256, 3072)	2362368	['layer_normalization_67[0][0]']
dropout_84 (Dropout)	(None, 256, 3072)	0	['dense_69[0][0]']
dense_70 (Dense)	(None, 256, 768)	2360064	['dropout_84[0][0]']
dropout_85 (Dropout)	(None, 256, 768)	0	['dense_70[0][0]']
add_67 (Add)	(None, 256, 768)	0	['dropout_85[0][0]', 'add_66[0][0]']
layer_normalization_68 (LayerNormalization)	(None, 256, 768)	1536	['add_67[0][0]']
multi_head_attention_34 (MultiHeadAttention)	(None, 256, 768)	28339968	['layer_normalization_68[0][0]', 'layer_normalization_68[0][0]']
add_68 (Add)	(None, 256, 768)	0	['multi_head_attention_34[0][0]', 'add_67[0][0]']
layer_normalization_69 (LayerNormalization)	(None, 256, 768)	1536	['add_68[0][0]']
dense_71 (Dense)	(None, 256, 3072)	2362368	['layer_normalization_69[0][0]']
dropout_86 (Dropout)	(None, 256, 3072)	0	['dense_71[0][0]']
dense_72 (Dense)	(None, 256, 768)	2360064	['dropout_86[0][0]']
dropout_87 (Dropout)	(None, 256, 768)	0	['dense_72[0][0]']

add_69 (Add)	(None, 256, 768)	0	['dropout_87[0][0]', 'add_68[0][0]']
layer_normalization_70 (LayerNormalization)	(None, 256, 768)	1536	['add_69[0][0]']
multi_head_attention_35 (MultiHeadAttention)	(None, 256, 768)	28339968	['layer_normalization_70[0][0]', 'layer_normalization_70[0][0]']
add_70 (Add)	(None, 256, 768)	0	['multi_head_attention_35[0][0]', 'add_69[0][0]']
layer_normalization_71 (LayerNormalization)	(None, 256, 768)	1536	['add_70[0][0]']
dense_73 (Dense)	(None, 256, 3072)	2362368	['layer_normalization_71[0][0]']
dropout_88 (Dropout)	(None, 256, 3072)	0	['dense_73[0][0]']
dense_74 (Dense)	(None, 256, 768)	2360064	['dropout_88[0][0]']
reshape_13 (Reshape)	(None, 16, 16, 768)	0	['add_65[0][0]']
dropout_89 (Dropout)	(None, 256, 768)	0	['dense_74[0][0]']
conv2d_transpose_21 (Conv2DTranspose)	(None, 32, 32, 512)	1573376	['reshape_13[0][0]']
add_71 (Add)	(None, 256, 768)	0	['dropout_89[0][0]', 'add_70[0][0]']
conv2d_72 (Conv2D)	(None, 32, 32, 512)	2359808	['conv2d_transpose_21[0][0]']
reshape_14 (Reshape)	(None, 16, 16, 768)	0	['add_71[0][0]']
batch_normalization_70 (BatchNormalization)	(None, 32, 32, 512)	2048	['conv2d_72[0][0]']
conv2d_transpose_20 (Conv2DTranspose)	(None, 32, 32, 512)	1573376	['reshape_14[0][0]']
re_lu_32 (ReLU)	(None, 32, 32, 512)	0	['batch_normalization_70[0][0]']
reshape_12 (Reshape)	(None, 16, 16, 768)	0	['add_59[0][0]']
concatenate_16 (Concatenate)	(None, 32, 32, 1024)	0	['conv2d_transpose_20[0][0]', 're_lu_32[0][0]']
conv2d_transpose_23 (Conv2DTranspose)	(None, 32, 32, 256)	786688	['reshape_12[0][0]']
conv2d_73 (Conv2D)	(None, 32, 32, 512)	4719104	['concatenate_16[0][0]']
conv2d_75 (Conv2D)	(None, 32, 32, 256)	590080	['conv2d_transpose_23[0][0]']
batch_normalization_71 (BatchNormalization)	(None, 32, 32, 512)	2048	['conv2d_73[0][0]']
batch_normalization_73 (BatchNormalization)	(None, 32, 32, 256)	1024	['conv2d_75[0][0]']
re_lu_33 (ReLU)	(None, 32, 32, 512)	0	['batch_normalization_71[0][0]']
re_lu_35 (ReLU)	(None, 32, 32, 256)	0	['batch_normalization_73[0][0]']
reshape_11 (Reshape)	(None, 16, 16, 768)	0	['add_53[0][0]']
conv2d_74 (Conv2D)	(None, 32, 32, 512)	2359808	['re_lu_33[0][0]']
conv2d_transpose_24 (Conv2DTranspose)	(None, 64, 64, 256)	262400	['re_lu_35[0][0]']
conv2d_transpose_26 (Conv2DTranspose)	(None, 32, 32, 128)	393344	['reshape_11[0][0]']
batch_normalization_72 (BatchNormalization)	(None, 32, 32, 512)	2048	['conv2d_74[0][0]']
conv2d_76 (Conv2D)	(None, 64, 64, 256)	590080	['conv2d_transpose_24[0][0]']
conv2d_79 (Conv2D)	(None, 32, 32, 128)	147584	['conv2d_transpose_26[0][0]']
re_lu_34 (ReLU)	(None, 32, 32, 512)	0	['batch_normalization_72[0][0]']
batch_normalization_74 (BatchNormalization)	(None, 64, 64, 256)	1024	['conv2d_76[0][0]']
batch_normalization_77 (BatchNormalization)	(None, 32, 32, 128)	512	['conv2d_79[0][0]']
conv2d_transpose_22 (Conv2DTranspose)	(None, 64, 64, 256)	524544	['re_lu_34[0][0]']
re_lu_36 (ReLU)	(None, 64, 64, 256)	0	['batch_normalization_74[0][0]']
re_lu_39 (ReLU)	(None, 32, 32, 128)	0	['batch_normalization_77[0][0]']
concatenate_17 (Concatenate)	(None, 64, 64, 512)	0	['conv2d_transpose_22[0][0]', 're_lu_36[0][0]']
conv2d_transpose_27 (Conv2DTranspose)	(None, 64, 64, 128)	65664	['re_lu_39[0][0]']

conv2d_77 (Conv2D)	(None, 64, 64, 256)	1179904	['concatenate_17[0][0]']
conv2d_80 (Conv2D)	(None, 64, 64, 128)	147584	['conv2d_transpose_27[0][0]']
batch_normalization_75 (BatchNormalization)	(None, 64, 64, 256)	1024	['conv2d_77[0][0]']
batch_normalization_78 (BatchNormalization)	(None, 64, 64, 128)	512	['conv2d_80[0][0]']
re_lu_37 (ReLU)	(None, 64, 64, 256)	0	['batch_normalization_75[0][0]']
re_lu_40 (ReLU)	(None, 64, 64, 128)	0	['batch_normalization_78[0][0]']
conv2d_78 (Conv2D)	(None, 64, 64, 256)	590080	['re_lu_37[0][0]']
conv2d_transpose_28 (Conv2DTranspose)	(None, 128, 128, 128)	65664	['re_lu_40[0][0]']
batch_normalization_76 (BatchNormalization)	(None, 64, 64, 256)	1024	['conv2d_78[0][0]']
conv2d_81 (Conv2D)	(None, 128, 128, 128)	147584	['conv2d_transpose_28[0][0]']
re_lu_38 (ReLU)	(None, 64, 64, 256)	0	['batch_normalization_76[0][0]']
batch_normalization_79 (BatchNormalization)	(None, 128, 128, 128)	512	['conv2d_81[0][0]']
conv2d_transpose_25 (Conv2DTranspose)	(None, 128, 128, 128)	131200	['re_lu_38[0][0]']
re_lu_41 (ReLU)	(None, 128, 128, 128)	0	['batch_normalization_79[0][0]']
concatenate_18 (Concatenate)	(None, 128, 128, 256)	0	['conv2d_transpose_25[0][0]', 're_lu_41[0][0]']
conv2d_82 (Conv2D)	(None, 128, 128, 128)	295040	['concatenate_18[0][0]']
reshape_10 (Reshape)	(None, 256, 256, 3)	0	['input_5[0][0]']
batch_normalization_80 (BatchNormalization)	(None, 128, 128, 128)	512	['conv2d_82[0][0]']
conv2d_84 (Conv2D)	(None, 256, 256, 64)	1792	['reshape_10[0][0]']
re_lu_42 (ReLU)	(None, 128, 128, 128)	0	['batch_normalization_80[0][0]']
batch_normalization_82 (BatchNormalization)	(None, 256, 256, 64)	256	['conv2d_84[0][0]']
conv2d_83 (Conv2D)	(None, 128, 128, 128)	147584	['re_lu_42[0][0]']
re_lu_44 (ReLU)	(None, 256, 256, 64)	0	['batch_normalization_82[0][0]']
batch_normalization_81 (BatchNormalization)	(None, 128, 128, 128)	512	['conv2d_83[0][0]']
conv2d_85 (Conv2D)	(None, 256, 256, 64)	36928	['re_lu_44[0][0]']
re_lu_43 (ReLU)	(None, 128, 128, 128)	0	['batch_normalization_81[0][0]']
batch_normalization_83 (BatchNormalization)	(None, 256, 256, 64)	256	['conv2d_85[0][0]']
conv2d_transpose_29 (Conv2DTranspose)	(None, 256, 256, 64)	32832	['re_lu_43[0][0]']
re_lu_45 (ReLU)	(None, 256, 256, 64)	0	['batch_normalization_83[0][0]']
concatenate_19 (Concatenate)	(None, 256, 256, 128)	0	['conv2d_transpose_29[0][0]', 're_lu_45[0][0]']
conv2d_86 (Conv2D)	(None, 256, 256, 64)	73792	['concatenate_19[0][0]']
batch_normalization_84 (BatchNormalization)	(None, 256, 256, 64)	256	['conv2d_86[0][0]']
re_lu_46 (ReLU)	(None, 256, 256, 64)	0	['batch_normalization_84[0][0]']
conv2d_87 (Conv2D)	(None, 256, 256, 64)	36928	['re_lu_46[0][0]']
batch_normalization_85 (BatchNormalization)	(None, 256, 256, 64)	256	['conv2d_87[0][0]']
re_lu_47 (ReLU)	(None, 256, 256, 64)	0	['batch_normalization_85[0][0]']
conv2d_88 (Conv2D)	(None, 256, 256, 3)	195	['re_lu_47[0][0]']

==========

Total params: 416,223,043

Trainable params: 416,216,131

Non-trainable params: 6,912

Anexo C: Inferencias adicionales de los modelos

Image Label U-Net UNETR

Image Label U-Net UNETR

Image Label U-Net UNETR

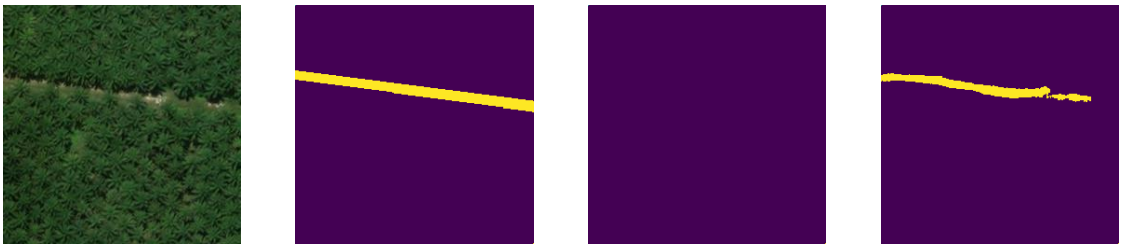


Image Label U-Net UNETR

Image Label U-Net UNETR

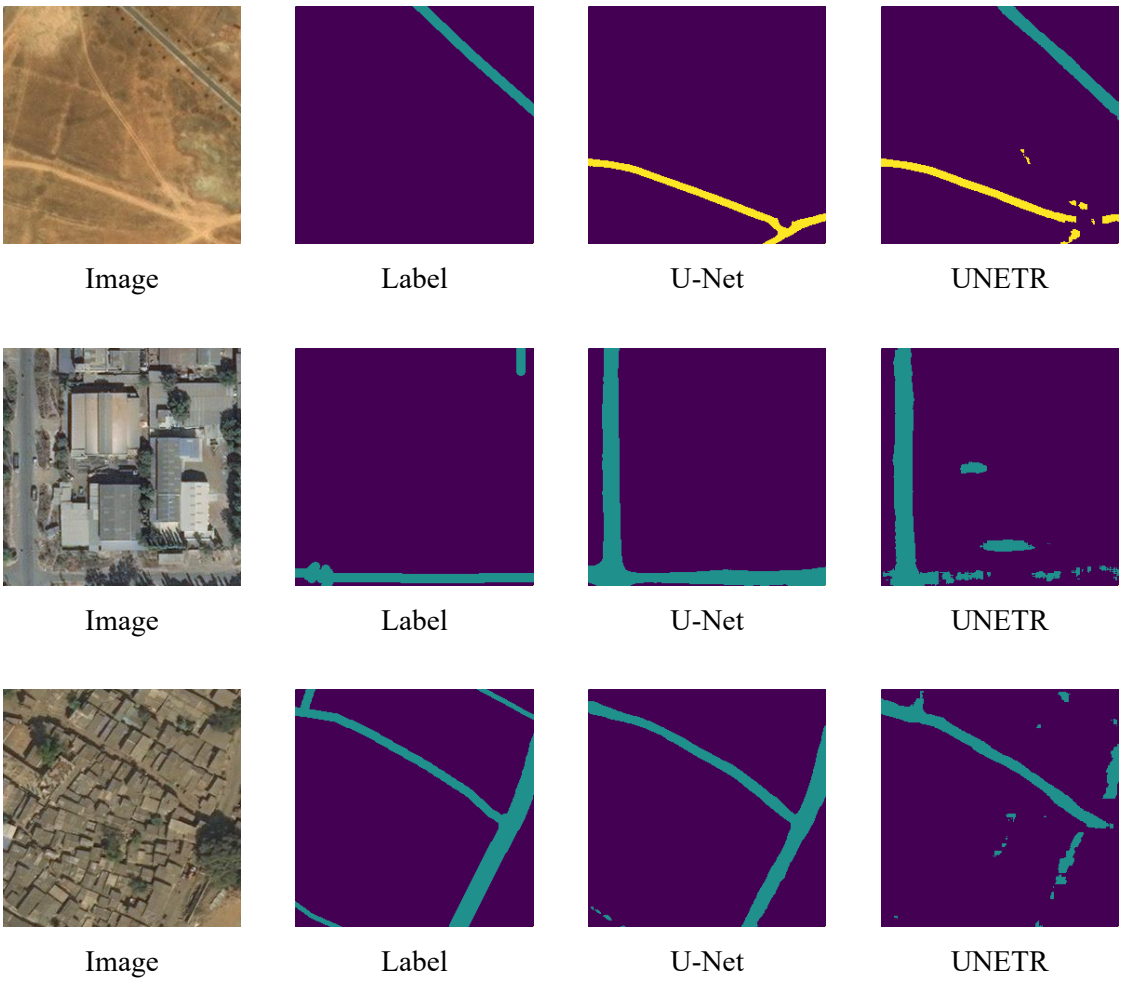


Figura 22. Inferencias adicionales sobre modelos.

Anexo D: Inferencia de los modelos sobre una imagen de dimensiones 1024x1024

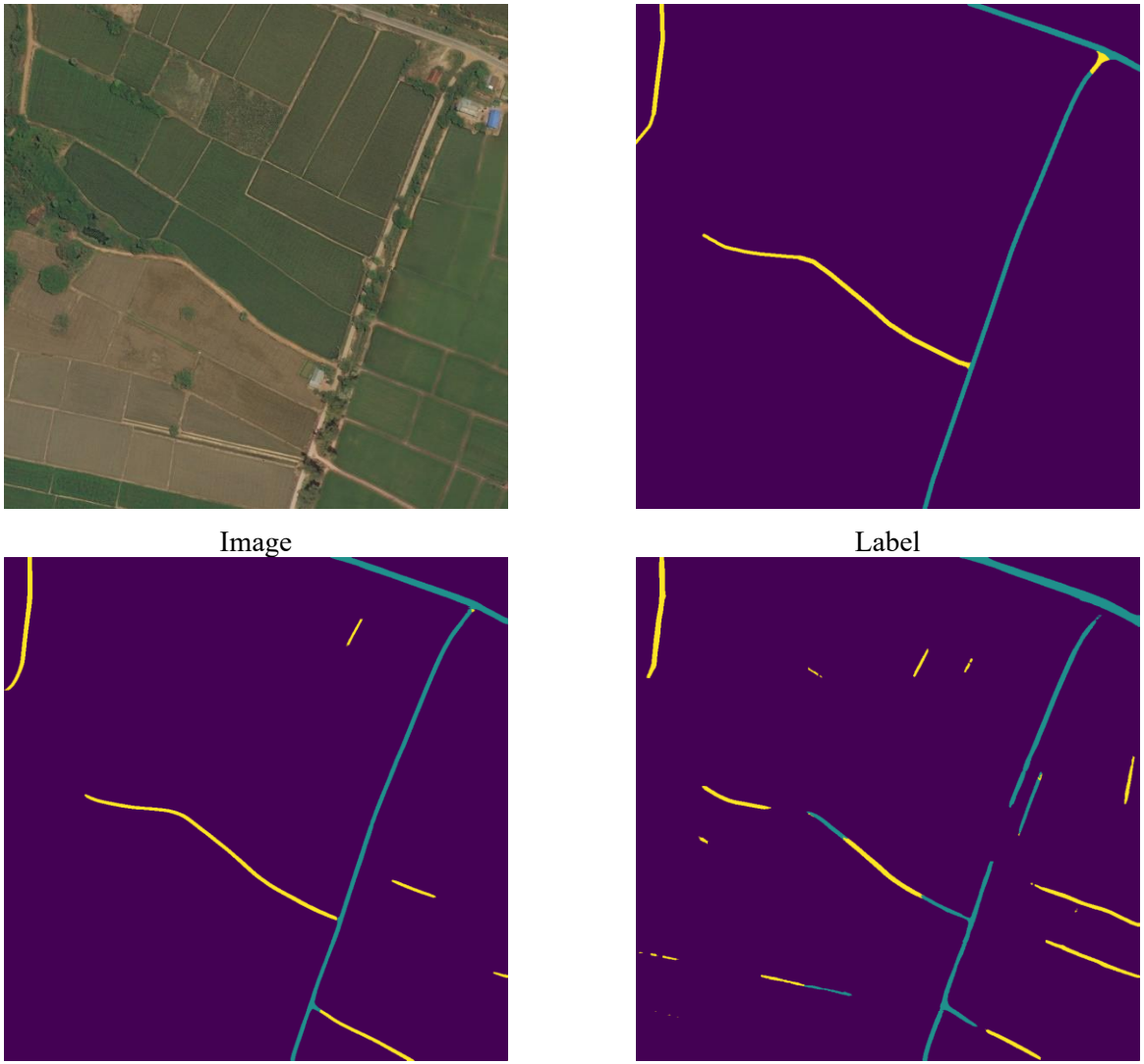


Figura 23. Inferencias adicionales sobre modelos sobre imágenes de 1024x1024.