ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA ELECTRÓNICA Y AUTOMATIZACIÓN

"DISEÑO DE UN ALGORITMO PARA DETERMINAR EL ÁREA DE RECURSOS HÍDRICOS EMPLEANDO PROCESAMIENTO DIGITAL DE IMÁGENES Y REDES NEURONALES"

TRABAJO DE TITULACIÓN

Tipo: Proyecto Integrador

Presentado para optar al grado académico de:

INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

AUTOR: CHRISTIAN ALEXANDER NÚÑEZ SEGOVIA

DIRECTOR: Ing. JOSÉ LUIS TINAJERO LEÓN

Riobamba – Ecuador

2022

Yo, Christian Alexander Núñez Segovia, declaro que el presente trabajo de titulación es de mi autoría y los resultados del mismo son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor asumo la responsabilidad legal y académica de los contenidos de este trabajo de titulación; el patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 01 de febrero de 2022

Christian Alexander Núñez Segovia
180472910-9

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA ELECTRÓNICA Y AUTOMATIZACIÓN

El Tribunal del Trabajo de Titulación certifica que: El trabajo de titulación; tipo: Propuesta Tecnológica, **DISEÑO DE UN ALGORITMO PARA DETERMINAR EL ÁREA DE RECURSOS HÍDRICOS EMPLEANDO PROCESAMIENTO DIGITAL DE IMÁGENES Y REDES NEURONALES**, realizado por el señor: **CHRISTIAN ALEXANDER NÚÑEZ SEGOVIA**, ha sido minuciosamente revisado por los Miembros del Tribunal del trabajo de titulación, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal Autoriza su presentación.

	FIRMA	**FECHA**
Ing. Jorge Luis Paucar Samaniego **PRESIDENTE DEL TRIBUNAL**	JORGE LUIS PAUCAR SAMANIEGO He revisado este documento	2022-03-15
Ing. José Luis Tinajero León **DIRECTOR DE TRABAJO DE TITULACIÓN**	Firmado electrónicamente por: **JOSE LUIS TINAJERO**	2022-03-15
Ing. Mayra Alejandra Pacheco Cúnduri **MIEMBRO DEL TRIBUNAL**	Firmado electrónicamente **MAYRA ALEJANDRA PACHECO CUNDURI**	2022-03-15

DEDICATORIA

A mis padres Carlos y Elizabeth y a mi hermano Johnn, que hicieron todo lo posible para que yo pudiera llevar a cabo mis estudios, pudiendo así prepararme día tras día durante mi formación profesional y nunca me faltase nada, a toda mi familia y amigos que, de alguna manera supieron brindarme su apoyo y buenos deseos.

Christian

AGRADECIMIENTO

Agradezco infinitamente a mis padres, gracias a su sacrificio y esfuerzo permitieron labrarme una carrera profesional, gracias por todo su apoyo y por velar por mi bienestar en todo momento, agradezco a mi hermano por impulsar mis talentos y por facilitarme los medios que hicieron de esta trayectoria más amena.

Un especial agradecimiento al Ing. José Luis Tinajero León, quien fue el proponente e impulsor del tema del presente trabajo, también agradezco a todos los docentes que conforman la carrera de Electrónica y Automatización que se esfuerzan día tras día para formar profesionales de bien.

Finalmente agradezco a todos mis compañeros de la facultad, a mis amigos y familiares, su amistad fue imprescindible en los buenos y malos momentos, gracias por todas las experiencias vividas y por no abandonarme cuando más los necesitaba.

Christian

TABLA DE CONTENIDO

CAPÍTULO II

CAPÍTULO III

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE ANEXOS

ÍNDICE DE ABREVIATURAS

IA	Inteligencia Artificial
LiDAR	Detección y localización de imágenes láser
GMES	Vigilancia mundial del medio ambiente y la seguridad
EU	Unión Europea
ESA	Agencia Espacial Europea
EUMETSAT	Organización Europea para la Explotación de Satélites
VRE	Borde rojo de la vegetación
NIR	Infrarrojo cercano
SWIR	Infrarrojos de onda corta
GSD	Distancia de muestra de tierra
ANN	Red Neuronal Artificial
CNN	Red Neuronal Convolucional
LBP	Patrones binarios locales
HOG	Histogramas de Gradientes Orientados
SIFT	Transformación de características invariantes de escala
IoT	Internet de las Cosas
UAV	Vehículo aéreo sin nombre
GCS	Almacenamiento en la nube de Google
GEE	Motor de Google Earth
API	Interfaz de Programación de Aplicaciones
GUI	Interfaz Gráfica de Usuario
GIS	Sistema de Información Geográfica
PDI	Procesamiento Digital de Imágenes
CIE	Comisión Internacional de Iluminación
ROI	Región de Interés
IOU	Intersección sobre Unión

RESUMEN

El presente trabajo tuvo como finalidad la segmentación y cálculo de áreas de recursos hídricos mediante el uso de Redes Neuronales Artificiales (RNA) y Procesamiento Digital de Imágenes (PDI). Se recurrió al uso de Inteligencia Artificial (IA), que es la ciencia que abarca las ramas empleadas para conseguir el objetivo planteado en este trabajo. Se empleó una Red Neuronal Convolucional (RNC), la cual tuvo que previamente ser entrenada con imágenes satelitales de recursos hídricos, obtenidas del portal web *Sentinel Hub* y de su herramienta *EO Browser*, las mismas recibieron un preprocesamiento para posteriormente ser etiquetadas. Una vez entrenada la red, se aplicó un postprocesamiento a las imágenes mediante Operaciones Morfológicas y Contornos Activos. Finalmente se realizó el cálculo del área del recurso mediante conteo de pixeles de la segmentación final obtenida, todo el proceso de cálculo fue implementado en una Interfaz Gráfica de Usuario (GUI) en el software Matlab. Los resultados obtenidos concluyeron en una alta precisión de segmentación por parte del software generado, superando una precisión del 95%, el software pudo procesar imágenes satelitales tomadas en diferentes espectros manteniendo su alta precisión. El software es capaz de realizar los cálculos de forma autónoma, por lo que posibilita el procesamiento masivo de datos, se recomienda trabajar en el espectro Infrarrojo de Onda Corta (SWIR) para obtener mejores resultados.

Palabras clave: <INTELIGENCIA ARTIFICIAL>, <REDES NEURONALES CONVOLUCIONALES>, <PROCESAMIENTO DIGITAL DE IMÁGENES>, <RECURSOS HÍDRICOS>, <OPERACIONES MORFOLÓGICAS>, <CONTORNOS ACTIVOS>, <IMÁGENES SATELITALES>, <BINARIZACIÓN>.

ABSTRACT

The purpose of this work was the segmentation and calculation of areas of water resources using Artificial Neural Networks (ANN) and Digital Image Processing (DIP). The use of Artificial Intelligence (AI) was used, which is the science that encompasses the branches used to achieve the objective set out in this work. A Convolutional Neural Network (CNN) was used, which had to be previously trained with satellite images of water resources, obtained from the Sentinel Hub web site and its EO Browser tool, these were pre-processed to later be labeled. Once the network was trained, a post-processing was applied to the images through Morphological Operations and Active Contours. Finally, the calculation of the area of the resource was carried out by counting pixels of the final segmentation obtained, the entire calculation process was implemented in a Graphical User Interface (GUI) in Matlab software. The results obtained by this software demonstrated a high precision of segmentation, exceeding a precision of 95%; the software was able to process satellite images taken in different spectra maintaining its high precision. It can do the calculations autonomously, therefore, it enables massive data processing, it is recommended to work in the Short-Wave Infrared (SWIR) spectrum for best results.

Keywords: <ARTIFICIAL INTELLIGENCE>, < CONVOLUTIONAL NEURAL NETWORKS >, <DIGITAL IMAGE PROCESSING>, <WATER RESOURCES>, <MORPHOLOGICAL OPERATIONS>, <ACTIVE CONTOURS>, <SATELLITE IMAGES>, <BINARIZATION>.

INTRODUCCIÓN

En los últimos años, los avances tecnológicos han propiciado la mejora y desarrollo de inteligencias artificiales (IA), con el propósito de imitar la inteligencia humana y otorgarles la capacidad de llevar a cabo tareas o procesos que, con tecnologías precedentes prácticamente eran imposibles de ejecutar debido al alto costo computacional o al tiempo que se tardarían en completar, con el único fin de prescindir de la supervisión de un humano. Actualmente, las IA se emplean en gran variedad de tareas, sobre todo en aquellas que requieren el procesamiento de una ingente cantidad de datos, las mismas tienen la capacidad de tomar decisiones de manera autónoma de acuerdo con los resultados de los análisis que estas puedan llegar a determinar. La inteligencia artificial nos permitirá lograr casi de todo de manera más barata, rápida y eficiente, y revolucionará sectores como la conducción, los viajes, la atención médica, la educación, el comercio, la agricultura, las finanzas, las ventas y el marketing.

La visión artificial o visión por computador es una rama del *Machine Learning* que se ha popularizado en gran medida gracias a los avances en la inteligencia artificial, siendo una herramienta de gran versatilidad aplicable en gran diversidad de campos como: teledetección, conducción autónoma, control de calidad, etc. La visión artificial nos permite adquirir, procesar y analizar imágenes de cualquier entorno, con el fin de que un computador sea capaz de entender la información que está procesando. Su objetivo en sí es comprender lo que ocurre en el mundo real y así poder tomar decisiones según se requieran.

Uno de los problemas más comunes en la visión artificial y el procesamiento digital de imágenes (PDI), que también es la base de este trabajo, es la segmentación semántica, que implica tomar características extraídas de una imagen y etiquetar cada píxel de la imagen para identificar patrones, objetos y entidades similares, para clasificarlos y categorizarlos en diferentes clases o etiquetas. Este proceso será aplicado a imágenes satelitales que contengan cualquier tipo de recurso hídrico, con el objetivo de dividir a la imagen en secciones etiquetadas diferenciando al recurso hídrico de cualquier tipo de suelo que lo rodee.

En este trabajo, aplicamos la segmentación semántica mediante la implementación de un algoritmo de aprendizaje profundo capaz de clasificar cada píxel de una imagen satelital. Después de distinguir las diversas categorías o clases, se puede aislar el contenido de interés de la imagen en una máscara binaria, aplicar un post procesado basado en PDI y calcular su área.

ANTECEDENTES

La inteligencia artificial (IA) es un área de las ciencias computacionales que atrae una gran cantidad de investigadores, esto debido a su amplia gama de aplicaciones en el campo

tecnológico. Actualmente, se están invirtiendo grandes esfuerzos por descubrir mecanismo, algoritmos que intenten imitar y simular la inteligencia humana, tarea algo difícil, ya que no tenemos el conocimiento suficiente sobre el cerebro humano.

Las Ciencias Computacionales fueron quienes dieron pie al desarrollo de las primeras Inteligencias Artificiales. Los computadores son quienes hacen posible implementar las teorías de la IA, donde son modeladas y ejecutadas. Los algoritmos de Inteligencia Artificial en general son bastantes extensos y de no ser por los grandes avances de velocidad y memoria de la industria computacional no sería posible hacerlos funcionar.

Proporcionar de capacidades sensoriales a un computador es una tarea difícil, ya que se deben tomar en cuenta muchas variables, desde el microprocesador de la máquina, hasta los sensores, tarjetas de conversión y el ruido(interferencias) producido por el entorno. Sin embargo, en la actualidad existe un especial interés por otorgarle a un computador la capacidad de ver, ignorando a los demás sentidos del ser humano. La visión artificial es un campo con una gran variedad de aplicaciones, desde procesos industriales hasta la astronomía, medicina, meteorología, etc.

La visión artificial es un campo de la inteligencia artificial que incluye todos los procesos y elementos que hacen posible 'ver' a las máquinas. La visión artificial o comprensión de imágenes representa la inferencia automática de la estructura del mundo a partir de una o más de dos dimensiones. Las imágenes y los atributos del mundo tridimensional pueden ser dinámicos.

Las redes neuronales son otra forma de intentar imitar ciertas características del cerebro humano, como la capacidad de recordar hechos y conectarlos de alguna manera. Algunos problemas no pueden resolverse mediante algoritmos simples, pero estos problemas tienen una solución común: la experiencia, y la experiencia acumulada se puede utilizar para resolver el problema. Las redes neuronales permiten resolver estos problemas y otorgan a las máquinas la capacidad de aprender de sus errores para que puedan acumular experiencia y adquirir conocimientos por sí mismas, son una forma sencilla de imitar los mecanismos del cerebro humano. Las redes neuronales son un nuevo sistema para el procesamiento de información, y sus unidades básicas de procesamiento están inspiradas en las neuronas, las células básicas del sistema nervioso humano. (Marcos et al. 2006)

Se están implementando proyectos similares al presente trabajo, por ejemplo, el proyecto WOIS (*Water Observation and Information System*) o Sistema de Observación e Información Hídrica, es una herramienta de código abierto que utiliza EO Data (*Earth Observation Data*) para monitorear, evaluar e inventariar los recursos hídricos usando escasos recursos económicos. El Sistema de Observación e Información Hídrica se desarrolló en el marco del proyecto TIGER-

NET, un componente clave de la iniciativa TIGER llevado a cabo por la Agencia Espacial Europea (ESA). Su objetivo central es apoyar las capacidades de monitoreo de recursos hídricos en África.

En el presente documento se plantea el desarrollo de un algoritmo que tenga la capacidad de reconocer y calcular áreas de recursos hídricos de imágenes satelitales de cualquier parte del mundo, haciendo uso de Inteligencia Artificial, mediante el desarrollo de un GUI en Matlab, en el cual se hará uso de redes neurales y visión artificial, todo esto con el fin realizar un estudio de la evolución de dicho recurso con el paso del tiempo.

PLANTEAMIENTO DEL PROBLEMA

¿Cómo determinar el área de recursos hídricos de fotografías satelitales mediante el uso de Visión Artificial y Redes Neuronales implementados en un GUI de Matlab?

JUSTIFICACIÓN TEÓRICA

Hoy en día, la proliferación de inteligencias artificiales ha revolucionado el mundo de la tecnología suponiendo un gran paso en la evolución computacional. Prácticamente estamos rodeados de las misma, ya que las IA hacen posible un sin número de procesos de los cuales hacemos uso diariamente, desde el teclado predictivo en nuestros smartphones, la detección de rostros en nuestras cámaras, hasta el piloto automático en vehículos autónomos. Sin duda la IA está en su máximo apogeo y no cabe duda de que su desarrollo se perpetuará hasta su perfeccionamiento.

En la actualidad existen varios softwares que se especializan en el campo de los Sistemas de Información Geográfica (SIG) como lo son: ArcGIS, que es de pago y QGIS que es de licencia gratuita, los cuales poseen diversas herramientas para el análisis de información geográfica en general, pero no se centran en los recursos hídricos como tal, también podemos mencionar a la herramienta WOIS que realiza el monitoreo de recursos hídricos en África. Es aquí donde se propone el desarrollo de un algoritmo que cuente con la función de procesamiento digital de imágenes y redes neuronales implementados en un GUI en Matlab, en el cual se realizarán un conjunto de procesamientos sobre una imagen satelital. La aplicación tendrá la capacidad de detectar recursos hídricos en la imagen que, luego de un conjunto de procesos se podrá hacer el reconocimiento y posterior binarización del área de interés de la cual se calculará su área en metros cuadrados.

Una vez comprobada la fiabilidad de este algoritmo con la ayuda de otros softwares especializados en SIG, se procederá a utilizar la información arrojada por dicho algoritmo para la

creación de un historial de áreas del recurso hídrico en el transcurso del tiempo, con la finalidad de hacer un estudio sobre estos datos y analizar el comportamiento del mismo con el paso de los años, dicho análisis nos permitirá determinar si el recurso hídrico está ganando o perdiendo cantidades considerables de agua, producto del cambio climático que está atravesando el planeta tierra.

El GUI no solo serviría para realizar el estudio de recursos hídricos, dependiendo de cómo se entrene a la red neuronal, será posible hacer el mismo estudio pero en glaciares.

JUSTIFICACIÓN APLICATIVA

La realización de este algoritmo comprende varias áreas de la Inteligencia Artificial: Visión artificial y redes neuronales, por lo que supone una investigación estudio y comprensión de estas ramas, que a la vez son un conjunto de procesos matemáticos y procesos computacionales. Dicho algoritmo será implementado en un GUI en Matlab, por lo que se deberá de tener conocimiento de este lenguaje de programación.

El algoritmo podrá ser usado para diferentes ámbitos investigativos que se requieran hacer sobre la superficie terrestre tanto a nivel local como global.

OBJETIVO GENERAL

Diseñar un algoritmo para la determinación del área de recursos hídricos empleando procesamiento digital de imágenes y redes Neuronales.

OBJETIVOS ESPECÍFICOS

- Revisar los métodos empleados en Visión Artificial para procesar y analizar imágenes, y entender como una red neuronal aprende y se modifica automáticamente.
- Analizar los métodos que se usan en Visión Artificial para la detección de patrones.
- Desarrollar la recopilación de datos para que la Visión Artificial detecte un objeto en específico.
- Realizar el entrenamiento de una red neuronal para que sea capaz de determinar un área en específico presente en una captura satelital.
- Evaluar la eficiencia del algoritmo desarrollado comparándolo con otros softwares de similares características.

CAPITULO I

1. MARCO TEÓRICO

En esta sección se abordarán temáticas como la observación de la tierra, los programas espaciales y los satélites empleados para el permanente monitoreo del planeta, así como también la teledetección en sí. También se tratan temas como la inteligencia artificial (IA), el *Machine Learning*, *Deep Learning* y las Redes Neuronales y, por último pero no menos importante, se realiza la descripción de cómo funciona el Procesamiento Digital de Imágenes.

1.1 Estado del Arte

ARTÍCULOS INTERNACIONALES

- **"Segmentación semántica de imágenes aéreas para la clasificación de cultivos"**

El objetivo de este trabajo se centra en el reconocimiento de los diferentes usos que se le pueden dar al terreno a través de su análisis por medio de imágenes aéreas, para lograrlo se aplicarán técnicas de visión artificial que permitan procesar grandes cantidades de datos de forma rápida y automatizada.

Las Redes Neuronales Convolucionales nos permiten realizar tareas de clasificación en imágenes, cuya salida nos dará el etiquetado de cada píxel en una clase en particular, para lograrlo utilizaremos técnicas de segmentación. (Díaz Pedrayes 2021)

Es importante tener en cuenta la resolución de las imágenes ingresadas en la Red Neural, ya que en imágenes de baja resolución es difícil reconocer los tipos de cultivos con éxito, mientras que en imágenes de alta resolución la clasificación es más precisa.

- **"Evaluación del uso de redes neuronales convolucionales para clasificación de cultivos mediante imagen multiespectral"**

Esta investigación se llevó a cabo para desarrollar un algoritmo capaz de segmentar imágenes de satélite y comparar los resultados obtenidos con las técnicas tradicionales.

Según Gonzales D. (2019) para llevar a cabo esta tarea, en primer lugar se proporcionará al algoritmo los fundamentos teóricos del aprendizaje profundo, centrándose en las redes convolucionales. A continuación, se explicarán los métodos utilizados en la clasificación de los cultivos para que, tras su análisis, se describan las decisiones tomadas en el preprocesamiento de los datos, justificando cada una de ellas. Finalmente, se presentarán los resultados.

El objetivo principal de este documento fue conocer el funcionamiento de las Redes Neuronales Convolucionales (CNN) para lograr la segmentación semántica de los diferentes tipos de cultivos a través de imágenes satelitales multiespectrales. Los modelos de capas convolucionales de CNN han conseguido extraer características suficientemente generales de los cultivos como para poder predecir con un alto porcentaje de acierto. Otros métodos tradicionales de clasificación de cultivos basados en técnicas clásicas de aprendizaje automático no tienen esta capacidad de extraer suficiente información espacial y temporal de las imágenes. En cualquier otro proceso de aprendizaje profundo, lo ideal es tener los datos lo más claros posible, evitando así el sesgo en las diferentes clases a clasificar y evitando el porcentaje de la clase no etiquetada.

- **"Estimación de área glaciar utilizando Redes Neuronales Convolucionales U-Net en imágenes multiespectrales** S**entinel 2 en el glaciar** A**usangate"**

Este trabajo pretende probar el modelo U-Net para estimar el área de los glaciares en imágenes multiespectrales obtenidas del satélite Sentinel 2.

Caillahua C. (2019) propone utilizar Redes Neuronales Convolucionales (CNN) con arquitectura U-Net como método de cartografía en glaciares. Los algoritmos de Inteligencia Artificial como CNN necesitan datos de entrenamiento y validación, para lo cual es óptimo utilizar la infraestructura computacional de *Google Earth Engine* (GEE) para recolectar muestras de imágenes del satélite Sentinel 2A y 2B. Para evaluar el rendimiento del modelo, se eligieron 12 escenas de prueba de fechas que están fuera de entrenamiento desde 2016 hasta 2019 en el glaciar Ausangate.

El modelo U-Net calculó la superficie de los glaciares de forma satisfactoria, obteniendo una precisión del 97,88%, esta investigación ha demostrado que la CNN U-Net puede utilizarse para la cartografía de superficies de glaciares y también mostró un buen rendimiento en comparación con el método tradicional. Además, con la ayuda de la red CNN U-Net, el índice diferencial de nieve normalizado (NDSI), el modelo puede utilizarse para cualquier glaciar.

- **"Fusión de Datos para Segmentación Semántica en Aplicaciones Urbanas de Teledetección Aérea Usando Algoritmos de Aprendizaje Profundo"**

El trabajo presentado por Chicchón Apaza (2018) busca desarrollar un método de combinación de datos basado en algoritmos de *Deep Learning* para la segmentación semántica en aplicaciones de teledetección urbana de datos obtenidos de cámaras RGB, cámaras multiespectrales y sistemas de *Light Detection and Ranging* o *Laser Imaging Detection and Ranging* (LiDAR).

La metodología se resume en los siguientes puntos: Recogida y selección de los datos, se realizará la segmentación semántica de las imágenes en seis clases. El entrenamiento del modelo que cumplirá el objetivo de realizar una segmentación semántica de las imágenes. La evaluación del modelo que se llevará a cabo principalmente según las reglas del *Semantic Labeling Challenge* 2D de la ISPRS.

Este trabajo resume la investigación sobre diferentes enfoques de fusión de datos utilizando el aprendizaje profundo aplicado en el área de la teledetección urbana. Se realizó un entrenamiento de extremo a extremo y basado en la transferencia del aprendizaje. Los resultados reflejan la necesidad de entrenar con un tamaño de lote mayor para mejorarlos. Otra opción de mejora es el uso de pesos de redes preentrenadas más complejas como *Resnet-50* o *Inception*.

- **"Clasificación y mapeo automático de coberturas del suelo en imágenes satelitales utilizando Redes Neuronales Convolucionales"**

Este trabajo tiene como objetivo desarrollar y evaluar un método computacional basado en redes neuronales convolucionales (CNN) para la clasificación de tierras, definiendo categorías secundarias: bosque, áreas con vegetación herbácea y / o arbustiva, áreas abiertas, áreas con poca o nula vegetación y humedales continentales.

Suárez et al. (2017) proponen un método de aprendizaje automático basado en CNN de arquitectura tipo ConvNet para la clasificación automática de cobertura terrestre de imágenes Landsat 5TM. La CNN fue capacitada a través de la interpretación visual de imágenes de satélite, los expertos utilizaron estas imágenes para generar mapas de cobertura terrestre para el Parque Nacional Tuparro. El modelo de validación se ejecuta utilizando datos del mapa superpuesto de la Amazonía colombiana, creado por el Sistema de Información Ambiental de Colombia.

Finalmente, se entrena un modelo de aprendizaje de CNN para la clasificación de la cobertura terrestre. Esto nos permite distinguir entre tres tipos de bosques, áreas seminaturales y un tipo de superficie de agua.

ARTÍCULOS NACIONALES

- **"Sistema inteligente de aprendizaje automático mediante el uso de un vehículo aéreo no tripulado (UAV) para el monitoreo de Oidium (sphaerotheca pannosa) en el cultivo de rosas de exportación en el Ecuador"**

El objetivo de este trabajo fue implementar técnicas de aprendizaje automático en la extracción de datos de vehículos aéreos no tripulados (UAV) para monitorear el Oidio (sphaerotheca pannosa) en el cultivo de rosas de exportación.

Espín Velasco (2021) utilizó técnicas de aprendizaje automático para crear un sistema inteligente. En este caso, se utiliza una red neuronal convolucional (CNN) para generar un modelo de aprendizaje automático basado en la presencia de la enfermedad (oidio). El modelo es capaz de identificar enfermedades con un nivel de confianza superior al 80%. Además, este sistema inteligente tiene un dispositivo IoT que captura datos de temperatura y humedad relativa de un vehículo aéreo no tripulado (UAV). Al mismo tiempo, el dron captura imágenes y video del área monitoreada.

Las CNN han demostrado un alto rendimiento en la detección e interpretación de imágenes. En este estudio, el detector de objetos obtenido (map) tiene una precisión del 73,1%, pero requiere un alto costo computacional. Por otro lado, el uso de dispositivos IoT puede proporcionar comunicación de dispositivo a dispositivo y recopilación de datos en tiempo real, y el uso de drones puede reducir el tiempo necesario para recopilar datos, generalmente de forma manual.

- **"Uso de Redes Neuronales Artificiales y Computación en la Nube para clasificar la cobertura del suelo en territorio ecuatoriano"**

El objetivo de esta investigación fue clasificar la vegetación en el centro de Ecuador utilizando *TensorFlow* y *Keras* con la ayuda de redes neuronales artificiales. Estas herramientas de computación en la nube incluyen *Google Collaboratory* (*Google Colab*), *Google Cloud Storange* (GCS) y *Google Earth Engine* (GEE).

Para realizar esta tarea, Castelo Cabay et al. (2021) utilizaron imágenes satelitales Sentinel-2 1C descargadas de la plataforma GEE para obtener una imagen en mosaico con estos atributos y una fracción de nubes del 10%. Se obtiene un valor medio de cada mosaico para generar la imagen a clasificar. A continuación, determinamos cinco múltiplos para clasificar: agua, nieve, árboles, vegetación y no vegetación. Luego, los datos etiquetados se exportan a la base de datos de activos de GEE como una serie de funciones y se leen desde *TensorFlow* a través de la API de *Python.*

El método utilizado es relativamente factible en términos de costo y tiempo, y los resultados son confiables. Esto facilitará la clasificación futura en áreas de cobertura más amplia y diferente en el territorio de Ecuador, ahorrando tiempo y dinero de agencias gubernamentales o particulares. Recientemente, se ha promulgado un método para clasificar la cobertura terrestre utilizando redes neuronales artificiales combinadas con la plataforma GEE y ha mostrado resultados efectivos para

los cambios en la cobertura terrestre. Esta es una nueva opción para cubrir todo tipo de suelos o grandes superficies.

- **“Sistema de Procesamiento Digital de Imágenes Satelitales para Cálculo de Áreas de Interés”**

En este artículo, Tinajero León et al. (2019) proponen el Procesamiento Digital de Imágenes obtenidas de satelitales por medio de teledetección, a través de un análisis basado en técnicas de visión artificial, implementado en GUI de Matlab, en la cual se realizan diversas operaciones sobre la imagen. La aplicación permite la selección del área de interés por medio de una herramienta de recorte, que facilita el preprocesamiento del área objetivo, la imagen es descompuesta en sus diversas matrices de color, para luego proceder a la aplicación de operaciones lógicas y de relleno de contornos. Adicionalmente se realiza la segmentación de la imagen incluyendo procesos automáticos y manuales de la herramienta Matlab, en la etapa final se muestra el valor aproximado del área de estudio antes del procesamiento, conformado por la etapa de filtrado y detección de bordes.

A través de la implementación de algoritmos y técnicas de procesamiento de imágenes, se desarrolló una aplicación con una interfaz gráfica de fácil uso e interpretación de datos, que permite el procesamiento de imágenes satelitales modificando parámetros (umbral, filtros espaciales) para discriminar e identificar formas y áreas de interés que contengan recursos hídricos.

- **“Desarrollo de un método para la clasificación automatizada de imágenes Landsat 8 mediante Redes Neuronales Artificiales”**

En este estudio, Tituaña Jami (2018) propuso la integración de redes neuronales artificiales y el desarrollo de un método para la clasificación automática de imágenes Landsat 8 a partir del reconocimiento de clases puras de píxeles. El trabajo realizado en este estudio se logró utilizando la plataforma SEPAL, que permitió la adquisición automática de mosaicos L8 en el área de estudio en pocos minutos, con el preprocesamiento y calibración adecuados. Los resultados muestran que las capacidades de preprocesamiento de la gestión de datos satelitales que existen en la actualidad han mejorado en comparación con el estado del campo hace diez años. Gracias al uso de redes neuronales artificiales y coeficientes de correlación cruzada de píxeles de satélite, este gran progreso es posible. Se discutió su uso y diferenciación regional en la teledetección de bosques, agricultura y cuerpos de agua.

- **“Determinación de cambios de la cobertura arbórea usando imágenes satelitales Landsat 7ETM+A a través de redes neuronales artificiales en la parroquia Achupallas, cantón Alausí, provincia de Chimborazo”**

En este trabajo, Arévalo Daniel (2017) intentó identificar cambios en la cobertura arbórea en la parroquia Achupallas, cantón Alausí, provincia de Chimborazo, a través de redes neuronales artificiales utilizando imágenes satelitales Landsat 7ETM +.

El proceso de análisis de la cobertura arbórea se describe mediante una clasificación supervisada en ArcGis. La fuente principal de este estudio son tres imágenes de satélite Landsat 7ETM + para los años correspondientes. Estas imágenes han sido previamente redondeadas y corregidas geométricamente. Para el análisis y modelado de la tasa de cambio, se utilizan software de aplicación *Land Change Modeler* (LCM) y GIS para calcular la probabilidad de cambio de la cubierta forestal mediante un algoritmo basado en redes neuronales artificiales. La taxonomía de supervisión determina mejor el porcentaje de cobertura forestal y el estado de enfermedades de diferentes tipos, como cuerpos de agua, bosques, tierras baldías, pastizales, plantaciones y cultivos. Las redes neuronales artificiales ayudan a crear escenarios futuros y se han realizado una serie de cambios entre 1991, 2001 y 2011.

Los mapas predictivos se generan mediante el seguimiento de la clasificación e información auxiliar de ríos, carreteras, pueblos, pendientes, etc. en el área de estudio y deben ajustarse a condiciones similares al ejecutar el software para evitar errores. El análisis estadístico de los mapas de pronóstico mostró tendencias similares a las de 1991, 2001 y 2011.

1.2 Observación de la tierra

Desde la década de 1920, los programas de Observación de la Tierra, en un marco internacional, han utilizado sistemáticamente sensores remotos para diversos fines, como el monitoreo atmosférico, el estudio y la predicción del clima, y como herramienta para propósitos militares. El desarrollo tecnológico en esta área permite, hoy, contar con sistemas satelitales para la recolección de diferentes tipos de datos, que una vez procesados, son interpretados y utilizados como herramientas para la toma de decisiones. (Duarte, Aguilar y Méndez, 2010)

Las imágenes captadas por satélites desde el espacio hacen posible la Observación de la tierra mediante un proceso denominado “teledetección”, el cual ha alcanzado un alto nivel de desarrollo y difusión, y se posiciona como una herramienta fundamental para el conocimiento y seguimiento de fenómenos naturales y acciones antrópicas y, los impactos que producen en el planeta.

Las imágenes satelitales disponibles de forma gratuita cuentan con diferentes resoluciones espaciales, temporales, espectrales y radiométricas, permitiendo su uso en diversas áreas y con propósitos diferentes. En la actualidad, existen capturas satelitales que son bastante adecuadas para trabajar a diversas escalas, desde un nivel regional hasta una propiedad o lote.

A continuación veremos cuales son algunas de las plataformas satelitales empleadas para el monitoreo de la superficie terrestre y sus principales aplicaciones. Algunas de las características comunes entre plataformas mencionadas aquí son: 1) Utilizan sensores pasivos, esto significa que los sensores captan radiación reflejada por un cuerpo de una fuente ajena al satélite, 2) Poseen plataformas de libre acceso para adquisición de datos a través de la red y 3) Es necesario estar capacitados para la manipulación e interpretación adecuada de los datos provistos:

- Landsat 8
- Sentinel 2 A
- Aqua y Terra

Las imágenes captadas mediante teledetección poseen aplicaciones muy variadas. La captación completa o parcial de un territorio a través del tiempo y espacio y, la capacidad de hacer visible lo invisible, resultan ser aportes de suma importancia a la hora de generar nuevo conocimiento en diferente disciplinas. (Ferreyra, 2016)

1.3 Teledetección

La detección remota -mejor conocida como teledetección-, es la ciencia que se dedica a monitorear la superficie o corteza terrestre, sin la necesidad de hacer contacto con la misma, como se puede observar en la Figura 1-1, los satélites se encuentran orbitando el Planeta de tal forma que sus sensores alcanzan a escanear la zona de interés desde una gran altitud. Este procedimiento se logra capturando y almacenando la radiación electromagnética reflejada para que, luego de un procesado, análisis e interpretación de los datos obtenidos poder tomar decisiones.

Existen tecnologías de teledetección aerotransportadas, espaciales, terrestres y marítimas que utilizan una gran cantidad de sensores montados en varias plataformas. Estos sensores observan energía electromagnética, energía acústica, energía ultrasónica, energía sísmica y energía magnética para el monitoreo ambiental y la observación de la tierra. Los sensores para la teledetección se clasifican en tipos pasivos y activos.

Los sensores pasivos miden la distribución de energía que se obtiene naturalmente a través del proceso de transferencia de radiación. Estos sensores pueden usar diferentes tipos de radiómetros y espectrómetros para detectar la reflexión (luz visible), la radiación (infrarroja e infrarroja

térmica) y los componentes de microondas del espectro electromagnético, en la Figura 2-1 se pueden apreciar todas las longitudes de onda del mismo. Estos sensores pueden producir imágenes pancromáticas, RGB, infrarrojas, hiperespectrales y multiespectrales.

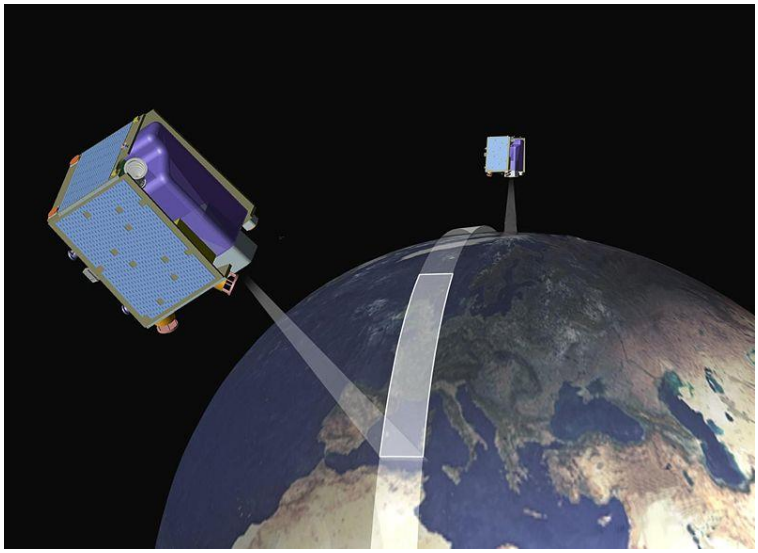


Figura 1-1. Teledetección realizada por un satélite.
Realizado por: Ingeoexpert, 2018.

Los sensores activos generan y envían energía para irradiar lo que se quiera captar. Esto significa que el sensor puede emitir su propia radiación hacia el objeto. El sensor detecta y mide la luz reflejada del objetivo aparente. Los sensores activos en los sistemas de detección remota incluyen radares, láseres o un LiDAR (*Light Detection and Ranging o Laser Imaging Detection and Ranging*) que en español significa: detección de luz y rango.

En los sistemas de teledetección, los resultados de las mediciones suelen cuantificarse y convertirse en imágenes digitales. Cada uno de los píxeles tienen un valor discreto en unidades digitales relacionado con la resolución espacial, espectral, radiométrica y/o temporal. Estas imágenes pueden contener defectos como la presencia de ruido debido a una variedad de factores, incluidas vibraciones anormales en el sistema de observación. Para abordar estos defectos, se requieren pasos de procesamiento adicionales. El objetivo principal es producir una mejor imagen para ayudar a visualizar o extraer la información.

Para poder interpretar y extraer información de imágenes teledetectadas se han desarrollado varias técnicas de procesamiento de imágenes. Las técnicas de preprocesamiento incluyen corrección atmosférica, geométrica y radiométrica, remuestreo, creación de mosaicos y relleno de espacios carentes de información suficiente. Además, se suelen aplicar cuatro técnicas avanzadas de

procesamiento de imágenes para mejorar aún más los atributos o la calidad de las imágenes de teledetección: mejora, restauración, transformación y segmentación. (Chicchón Apaza, 2018)

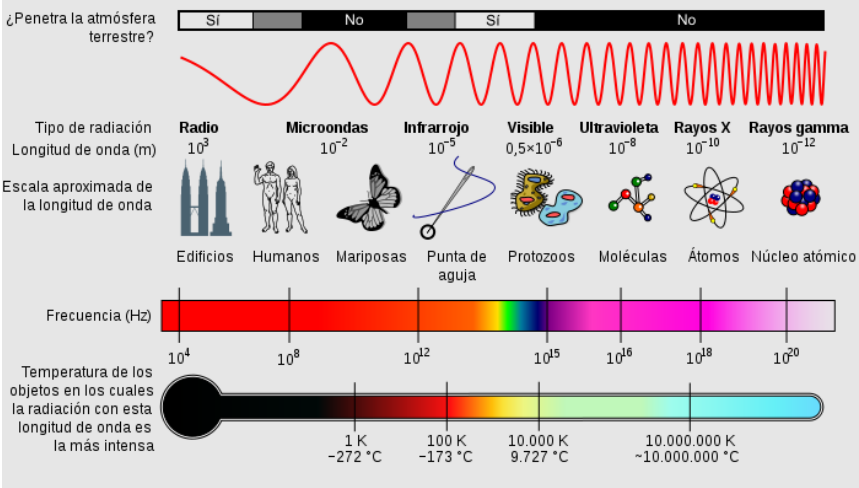


Figura 2-1. Espectro electromagnético.

Realizado por: Cosmicus, H. 2010.

1.3.1 Teledetección Satelital

Los satélites han abierto una serie de nuevas posibilidades en el mundo de la teledetección. La teledetección por satélite nos permite observar objetos desde mayores altitudes y distancias. También es un paso muy importante hacia sistemas que detectarán cambios en el clima e incluso conocimiento de nuestro propio planeta.

La teledetección satelital se realiza mediante satélites, por lo general, satélites meteorológicos. Los sistemas de teledetección que llevan a bordo estos dispositivos puestos en órbita son pasivos, ya que aprovechan la radiación reflejada por el objeto investigado para la recopilación de datos.

Este tipo de teledetección se ha utilizado durante décadas para observar una gran variedad de procesos que se producen en nuestro planeta, pero recientemente, se ha dado paso a la teledetección mediante drones. Este nuevo sistema tiene la ventaja de producir imágenes más precisas y detalladas debido al uso de cámaras de altas resoluciones, esta tecnología se usa de manera exclusiva para estudios que no requieran imágenes panorámicas o a grandes distancias. Por ejemplo, se utiliza para la observación aérea de tierras destinadas a cultivos. (Ingeoexpert 2018)

1.4 Programa Copernicus

El programa Copernicus es un proyecto de la Unión Europea (EU) para observar y monitorear nuestro planeta y a sus numerosos ecosistemas. Este programa se llamó inicialmente "*Global Monitoring for Environment and Security*" (GMES).

El programa utiliza una variedad de tecnologías para proporcionar datos comerciales y servicios de información gratuitos en una serie de áreas de aplicación, desde satélites en el espacio hasta sistemas de medición aérea, marítima y terrestre. Copernicus planea observar nuestro entorno, recopilar y almacenar datos, analizar y crear productos para una toma de decisiones efectivas.

El Servicio de Información de Copernicus es provisto por todos los datos generados por los satélites "Sentinel" (una constelación de seis familias de satélites) y la información de decenas de satélites de terceros (misiones participantes).

Copernicus guarda estos datos y nos ayuda a obtener una gran cantidad de información confiable y actualizada sobre el estado de nuestro planeta. Estos datos se utilizan para crear una variedad de productos, incluidos datos estadísticos y mapas topográficos. Estos datos se analizan para generar métricas útiles para que los investigadores y los usuarios finales informen sobre tendencias pasadas, presentes y futuras.

Copernicus no solo ha contribuido a la ciencia y la tecnología en Europa, sino que también ha integrado marcos de servicio público para que las personas puedan tener acceso libre, abierto y completo a los datos recopilados. Cualquiera puede hacer uso de esta información.

Las misiones fueron destinadas para tratar temas interrelacionados y se organizan en torno a seis ejes:

- Seguridad.
- Cambio climático
- Gestión de emergencias.
- Vigilancia medioambiental marina.
- Vigilancia terrestre.
- Vigilancia atmosférica.

La Agencia Espacial Europea (ESA) se dedica al desarrollo de la parte espacial del programa Copernicus y de la administración de los satélites Sentinel 1, Sentinel 2 y Sentinel 5P. La ESA también ejecuta la misión terrestre Sentinel 3.

La Organización Europea para la Explotación de Satélites (EUMETSAT) es responsable del funcionamiento de los satélites Sentinel 3 y de la ejecución de las misiones oceánicas. Además, se hará cargo de la administración y entrega de los productos de instrumentos Sentinel 4 y Sentinel 5 y los satélites Sentinel 6.

La ESA y EUMETSAT coordinarán la entrega de datos de los más de 30 satélites que participan en la misión. (Morales, 2020)

1.4.1 Sentinel 2

Sentinel 2 es una misión de observación de la Tierra desarrollada por la ESA para el proyecto Copernicus, utilizando dos satélites idénticos, Sentinel 2A y Sentinel 2B. La misión tiene como objetivo observar la Tierra y proporcionar servicios para monitorear la evolución de la vegetación, los cambios en la corteza y el manejo de desastres naturales. Las principales características de Sentinel 2 son:

- Imágenes multiespectrales de 13 bandas.
- Bandas en el espectro visible, infrarrojo cercano, infrarrojo de onda corta y de espectro electromagnético.
- Actualiza sus imágenes cada 5 días, manteniendo los mismos ángulos de visión.
- Dependiendo de la banda obtiene imágenes con una resolución espacial de 10, 20 o 60 metros por píxel.
- Su campo de visión es de 290 km.
- Política de libre acceso a sus datos.

Tabla 1-1: Información de las bandas de Sentinel 2A y Sentinel 2B.

Banda	Longitud de onda (nm)		Ancho de banda (nm)		GSD (m/píxel)
	S-2A	S-2B	S-2A	S-2B	
B1 Coastal aerosol	442,7	442,2	21	21	60
B2 Blue	492,4	492,1	66	66	10
B3 Green	559,8	559	36	36	10
B4 Red	664,6	664,9	31	31	10
B5 VRE	704,1	703,8	15	16	20
B6 VRE	740,5	739,1	15	15	20
B7 VRE	782,8	779,7	20	20	20
B8 NIR	832,8	832,9	106	106	10
B8A Narrow NIR	864,7	864	21	22	20
B9 Water vapour	945,1	943,2	20	21	60
B10 SWIR cirrus	1374	1376,9	31	30	60
B11 WIR	1614	1610,4	91	94	20
B12 SWIR	2202	2185,7	175	185	20

Fuente: Díaz Pedrayes, 2021.

Realizado por: Núñez, Christian, 2021.

En la Tabla 1-1 podemos apreciar todas las bandas que pueden captar los satélites Sentinel 2A y 2B, su longitud de onda, ancho de banda, así como también la resolución espacial con la que trabajan sus sensores.

Las siglas "VRE" significan "*Vegetation Red Edge*", "NIR" quiere decir "*Near InfraRed*", y finalmente "SWIR" es "*Short Wave InfraRed*". De color rojo se muestran las bandas con un GSD de 60 m/píxel, de amarillo las bandas de 20 m/píxel y de verde las de 10 m/píxel. (Díaz Pedrayes, 2021)

1.5 Inteligencia artificial

La inteligencia artificial (IA) es la capacidad de las máquinas, como los humanos, para utilizar algoritmos, aprender de los datos y utilizar el conocimiento adquirido para tomar decisiones. Sin embargo, a diferencia de los humanos, los dispositivos de inteligencia artificial no necesitan descanso y pueden analizar una gran cantidad de información a la vez. Además, cuando una máquina realiza la misma tarea que un humano, la tasa de error es mucho menor.

Las tecnologías basadas en IA se están utilizando para empoderar a los humanos a realizar mejoras significativas e incrementar la eficiencia en casi todas las áreas de la vida. Sin embargo, el crecimiento acelerado de la Inteligencia Artificial también requiere atención para prevenir y analizar las posibles desventajas directas o indirectas que podrían surgir del rápido crecimiento de esta. La inteligencia artificial se puede aplicar a casi cualquier situación. Estas son solo algunas de las aplicaciones tecnológicas de la IA que en la actualidad se están desarrollando rápidamente:

- Reconocimiento, clasificación y etiquetado de imágenes fijas.
- Mejorar la eficiencia de las estrategias de algoritmos comerciales.
- Procesamiento de datos de pacientes eficiente y escalable.
- Mantenimiento predictivo.
- Detección y clasificación de objetos.
- Distribución de contenido en redes sociales.
- Protección contra amenazas a la seguridad de la red.

La inteligencia artificial también puede proporcionar sugerencias y predicciones sobre temas importantes en nuestras vidas en diferentes áreas como: la salud, el bienestar, la educación, el trabajo y las relaciones interpersonales. También puede transformar la forma en que se llevan a cabo los negocios, proporcionando una ventaja competitiva a las empresas que buscan comprender y aplicar estas herramientas de manera rápida y eficiente. Otro beneficio de la inteligencia artificial es que permite que las máquinas y los robots realicen tareas que los humanos

encuentran difíciles, aburridas o peligrosas. Permite a las personas hacer cosas que antes pensaban eran imposibles. (Rouhiainen, 2018)

1.5.1 Aprendizaje Automático

El aprendizaje automático o *Machine Learning*, es una técnica de análisis de datos que crea modelos analíticos automáticamente. Esta es una rama de la inteligencia artificial, cuya esencia es que los sistemas pueden aprender de los datos, reconocer patrones y tomar decisiones con una mínima intervención humana. (Morocho Sande y Pulig Cesén, 2021)

1.5.2 Aprendizaje Profundo

El aprendizaje profundo o *Deep Learning*, es una capacidad de la inteligencia artificial que imita el funcionamiento del cerebro humano en el procesamiento de datos y la creación de modelos de decisión. El aprendizaje profundo es un subconjunto del *Machine Learning* en IA cuyas redes son capaces de aprender sin supervisión, basándose en datos no estructurados o sin etiquetar. También se conoce como aprendizaje neuronal profundo o redes neuronales profundas. (Morocho Sande y Pulig Cesén, 2021)

1.5.3 Modelo de predicción

El análisis predictivo es el uso de datos históricos, aprendizaje automático e inteligencia artificial para predecir lo que sucederá en el futuro. Estos datos históricos se introducen en un modelo matemático que tiene en cuenta las principales tendencias y patrones en los datos. (Morocho Sande y Pulig Cesén, 2021)

1.6 Machine Learning

El *Machine Learning* o aprendizaje automático, es un mecanismo mediante el cual una computadora aprende a resolver un problema por sí misma. En este caso, el programador diseña el algoritmo de aprendizaje apropiado para el problema que se va a resolver, pero la computadora resuelve el problema por sí misma utilizando los datos accedidos y la heurística de aprendizaje incorporada en el algoritmo creado por el programador. En cierto sentido, la computadora puede programarse por sí misma. (Berzal, 2018)

El aprendizaje automático es un concepto que solo se puede aplicar a máquinas con inteligencia artificial, estas máquinas suelen ser destinadas a realizar diagnósticos, predicciones e identificación. Estas máquinas a menudo aprenden de los datos proporcionados, los mismos que comúnmente son denominados datos de entrenamiento, que son datos de muestra o datos históricos que se utilizan para entrenar el sistema. Las máquinas aprenden a analizar patrones en los datos y utilizan dichos patrones para derivar resultados de análisis.

Existen varios mecanismos de aprendizaje para el entrenamiento de las máquinas. Los más utilizado de estos mecanismos son: el aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. (Dark, 2018a). La Figura 3-1 muestra los tres subconjuntos en los que más comúnmente se subdivide el aprendizaje automático.

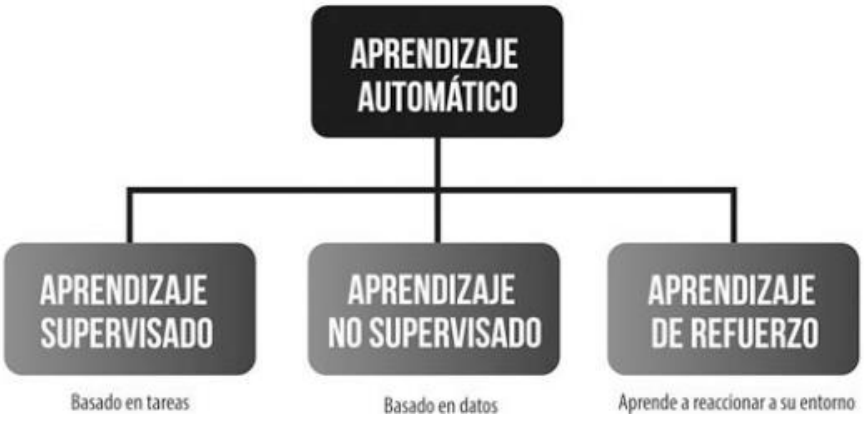


Figura 3-1. Tipos de Aprendizaje Automático.

Realizado por: Rouhiainen, L. 2018.

1.6.1 Aprendizaje Supervisado

En el aprendizaje supervisado, los ejemplos de aprendizaje van acompañados de la salida correcta que el sistema debería poder reproducir. Entrenar un modelo de aprendizaje supervisado es una cuestión de ajustar los parámetros del modelo para que estén lo más cerca posible del resultado deseado. Una vez entrenado el modelo, lo que realmente importa es si el modelo puede generalizar correctamente. Esta capacidad de generalizar significa que el modelo produce la salida adecuada para los datos de entrada que son diferentes a los datos utilizados durante el entrenamiento. Este método requiere de supervisión humana para poder proporcionar retroalimentación. (Berzal, 2018)

1.6.2 Aprendizaje no supervisado

El aprendizaje no supervisado crea explicaciones, hipótesis o teorías a partir de un conjunto de hechos y observaciones, sin información adicional sobre cómo clasificar los ejemplos en el conjunto de aprendizaje. El aprendizaje no supervisado determina cómo se deben agrupar los datos del conjunto de entrenamiento o qué patrones del conjunto de entrenamiento son más interesantes, por lo tanto, este método no requiere de intervención humana. (Berzal, 2018)

1.6.3 Aprendizaje mediante refuerzos

El aprendizaje por refuerzo es básicamente un mecanismo de experimentación iterativa donde los algoritmos aprenden de la experiencia. Los ensayos exitosos recompensarán la estrategia utilizada (refuerzo positivo). El fracaso penaliza la estrategia que condujo al fracaso (refuerzo negativo).

En el aprendizaje por refuerzo, la retroalimentación positiva obtenida al tomar medidas para lograr el objetivo actúa como una señal de supervisión. (Berzal, 2018)

1.7 Deep Learning

El *Deep Learning* o aprendizaje profundo es un método de aprendizaje automático que utiliza una arquitectura de red neuronal. La palabra "profundo" se refiere al número de capas de estas técnicas. Las redes neuronales tradicionales cuentan con una única capa, mientras que las redes neuronales profundas poseen más de una capa.

El aprendizaje profundo es un tipo de inteligencia artificial que imita la forma en que los humanos aprenden a adquirir cierto tipo de conocimiento. También es un método automatizado que se utiliza para el análisis predictivo. (Saez De La Pascua, 2019)

Inspirándose en el cerebro humano, los conexionistas han construido un modelo relativamente simple de computadora, que consta de múltiples unidades básicas llamadas neuronas. Estas neuronas (elementos de procesamiento) están conectadas para formar una red neuronal artificial. Al manipular las conexiones entre las neuronas de la red, puede hacer que la red neuronal funcione de la manera que se desee. (Berzal, 2018)

Los algoritmos de aprendizaje profundo se construyen en capas de complejidad y extracción cada vez mayores. Cada algoritmo que forma la jerarquía realiza transformaciones no lineales en las entradas y utiliza el conocimiento aprendido para crear modelos estadísticos como salidas. La iteración continúa hasta que la salida alcanza una precisión aceptable.

En el aprendizaje profundo, el modelo descubre la relación entre variables aprendiendo por sí mismo. En un modelo que utiliza el aprendizaje profundo, el propio modelo puede seleccionar características sin intervención humana. (Saez De La Pascua, 2019)

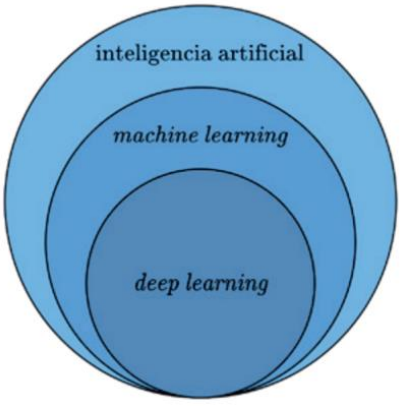


Figura 4-1. Relación entre Inteligencia Artificial, *Machine Learning* y *Deep Learning*.

Realizado por: Pérez, I.; Gegúndez, M. 2021.

En la Figura 4-1 se aprecia al *Deep Learning* como un subconjunto del *Machine Learning*, que a su vez es una de las disciplinas englobadas dentro del campo de la Informática conocido como Inteligencia Artificial.

1.7.1 Redes Neuronales Artificiales

Una Red Neuronal Artificial, del inglés, *artificial neural network* (ANN) es un paradigma de procesamiento de información inspirado en el funcionamiento del cerebro. Un elemento importante de este paradigma es su estructura. Una red neuronal artificial se compone de muchos elementos de procesamiento (neuronas) que trabajan en conjunto para resolver un problema específico. La red neuronal actual se basa en el modelo matemático de neuronas propuesto por McCulloch y Pitts en 1943. En tal modelo (ver Figura 5-1), cada neurona recibe un conjunto de entradas $\{x_1, x_2, \dots, x_D\}$ y devuelve una salida y. Además, en una ANN, existen muchas conexiones entre las distintas neuronas que la componen. Estas conexiones imitan las conexiones entre las células nerviosas del cerebro y, al igual que el cerebro, pueden establecerse con mayor o menor intensidad. Para las ANN, esta intensidad está determinada por los pesos sinápticos (o pesos, para abreviar). Por tanto, cada entrada x_i de una neurona está influenciada por el peso w_i. (Palma Méndez y Marín Morales, 2008)

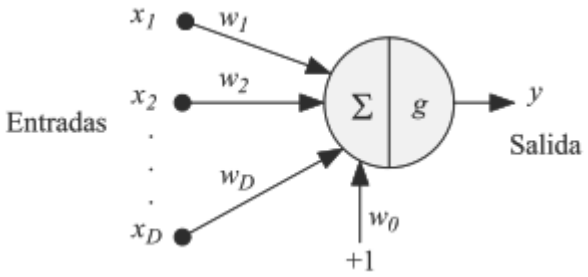


Figura 5-1. Modelo de neurona de McCulloch-Pitts.

Realizado por: Palma Méndez, J; Marín Morales, R. 2008.

El primer paso para obtener la salida neuronal y es calcular la suma ponderada a (Ecuación 1.1) de las entradas, llamada activación neuronal.

$$a = \sum_{i=1}^{D} w_i x_i + w_0 \qquad (1.1)$$

donde w_0 es el umbral o desviación utilizado para compensar la diferencia entre la media de las entradas en el conjunto de entrenamiento y la media correspondiente de la salida deseada. Luego, a partir de este valor a, aplicamos una función llamada función de activación o función de transferencia $g(a)$ (Ecuación 1.2) para obtener la salida y de la neurona, es decir:

$$y = g(a) = g\left(\sum_{i=1}^{D} w_i x_i + w_0\right) = \left(\sum_{i=0}^{D} w_i x_i\right) \tag{1.2}$$

donde, como se puede ver, asumiendo que la entrada adicional x_0 tiene un valor fijo de 1, el umbral w_0 puede considerarse como un peso adicional (ver Figura 5-1). Finalmente, si **w** es el vector de peso y **x** es el vector de entradas de red, entonces la ecuación se puede reescribir como una representación vectorial como $g(a) = g(\boldsymbol{w}^T \boldsymbol{x})$. La función de transferencia utilizada en este modelo básico de McCulloch-Pitts es la función escalonada definida por la Ecuación 1.3:

$$g(a) = \begin{cases} 0 & cuando \quad a < 0 \\ 1 & cuando \quad a \geq 0 \end{cases} \tag{1.3}$$

Se parece a la Figura 6-1 (a). Como veremos más adelante, el modelo actual opta por otro tipo de funciones, que suelen ser monótonas y derivables. La más común es la función lineal, como se muestra en la Figura 6-1 (b), y es representada por la Ecuación 1.4.

$$g(a) = a \tag{1.4}$$

La función sigmoidea (Figura 6-1 (c)), expresada por la Ecuación 1.5,

$$g(a) = \frac{1}{1 + e^{-a}} \tag{1.5}$$

La función tangente hiperbólica (Figura 6-1 (d)), que es representada por la Ecuación 1.6,

$$g(a) = tanh = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}} \tag{1.6}$$

Y la función gaussiana (Figura 6-1 (e)), cuya expresión es mostrada por la Ecuación 1.7,

$$g(a) = \exp\left(-\frac{(a-\mu)^2}{2\sigma^2}\right) \tag{1.7}$$

La cantidad de neuronas que componen una ANN y el método de conexión de las neuronas se denomina topología de red. (Palma Méndez y Marín Morales, 2008). En la Figura 6-1 se muestran de forma gráfica las funciones de activación más comunes.

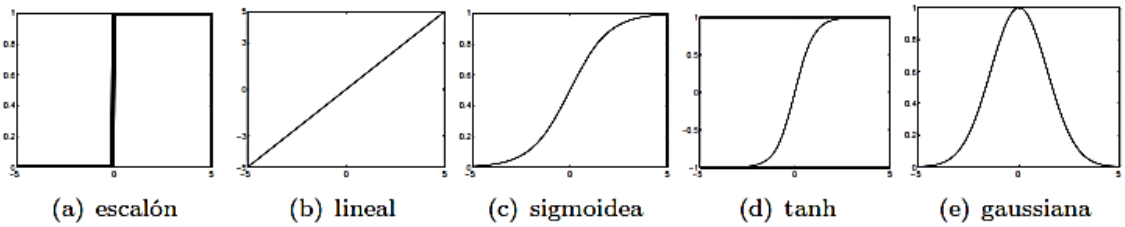


Figura 6-1. Funciones de activación o transferencia.

Realizado por: Palma Méndez, J; Marín Morales, R. 2008.

1.7.2 Elementos de una Red Neuronal

La ANN consta de un conjunto de elementos de procesamiento interconectados, entre los cuales se envía información a través de las conexiones. La Figura 7-1 muestra el diagrama esquemático básico de una Red Neuronal Artificial. Muestra las diferentes capas de la topología y tiene una estructura llamada *feed-forward* (hacia adelante) debido a la dirección del flujo de la información.

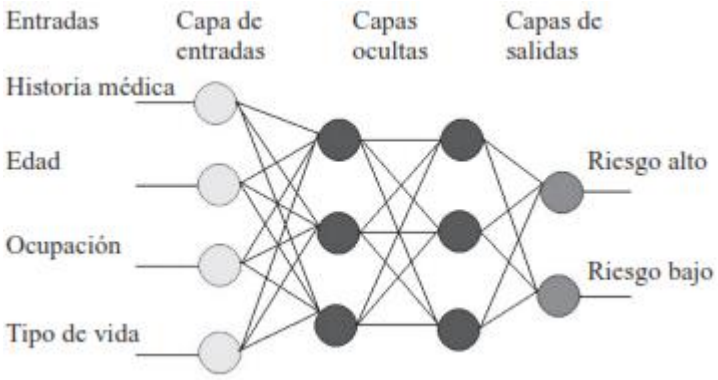


Figura 7-1. Modelo esquemático de una ANN.

Realizado por: Ponce Cruz, P. 2010.

Los elementos básicos de una ANN son:

- Un conjunto de unidades de procesamiento (neuronas).
- Conexiones entre las unidades (los pesos y valores están asociados con cada conexión).
- La función de salida o activación de cada unidad de procesamiento.

El modelo de neurona y la arquitectura de la red muestran cómo la red transforma las entradas en salidas. Los elementos computacionales individuales que componen el modelo de un sistema neuronal artificial se denominan neuronas artificiales. (Ponce Cruz, 2010)

1.7.3 Clasificación de las Redes Neuronales Artificiales

Las redes neuronales artificiales se clasifican en redes neuronales de una sola capa y redes neuronales multicapa según su estructura y topología de red.

1.7.3.1 Red Neuronal Monocapa - Perceptrón simple

El término Perceptrón fue acuñado por Frank Rosenblatt en 1960, donde se desarrolló ampliamente la arquitectura de la red neuronal. Un Perceptrón es un modelo inspirado en el de McCulloch y Pitts. En este modelo, cada neurona tiene asignada un peso fijo o de preprocesamiento. Los perceptrones se usan solo para el reconocimiento de patrones, pero también se pueden usar para otras funciones. (Dark, 2018b)

Un perceptrón es un algoritmo de clasificación binaria simple que divide un conjunto de señales de entrada en dos partes, "si" y "no". Los perceptrones son máquinas de aprendizaje muy simples porque pueden aceptar múltiples entradas y generar decisiones de salida con pesos que indican la importancia de cada entrada.

Como se puede apreciar en la Figura 8-1, un perceptrón simple tiene nodos con entradas interconectadas asociadas con un número dado, y estos nodos son responsables de calcular la salida esperada cuando se conecta al *kernel* y luego se pasa a la función de activación. (Morocho Sande y Pulig Cesén, 2021)

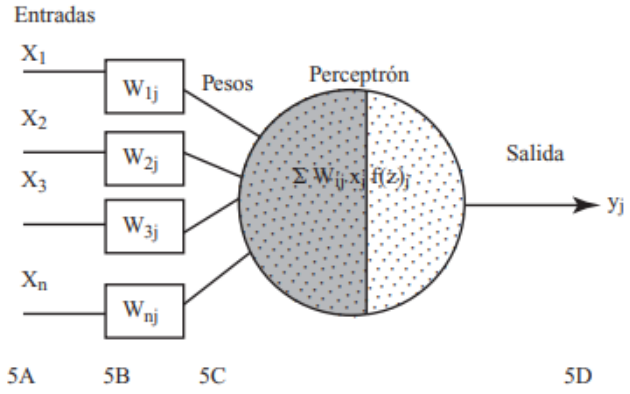


Figura 8-1. Estructura del Perceptrón.

Realizado por: Ponce Cruz, P. 2010.

1.7.3.2 Red Neuronal Multicapa - Perceptrón Multicapa

Los perceptrones multicapa, también conocidos como "redes neuronales hacia adelante", son redes más complejas que los perceptrones porque constan de múltiples capas de neuronas. Hay capas de entrada, ocultas y de salida, los perceptrones multicapa suelen estar completamente conectados. Esto significa que hay una conexión de cada perceptrón en una capa a cada perceptrón en la siguiente capa.

En la Figura 9-1, hay una capa oculta entre las capas de entrada y salida, que es el motor computacional real, porque las neuronas en esta red están divididas en múltiples capas. Dado que este tipo de red tiene varias capas, las conexiones entre neuronas son de tipo *feedforward*

(conexiones hacia adelante) y tipo *feedback* (conexiones hacia atrás). (Morocho Sande y Pulig Cesén, 2021)

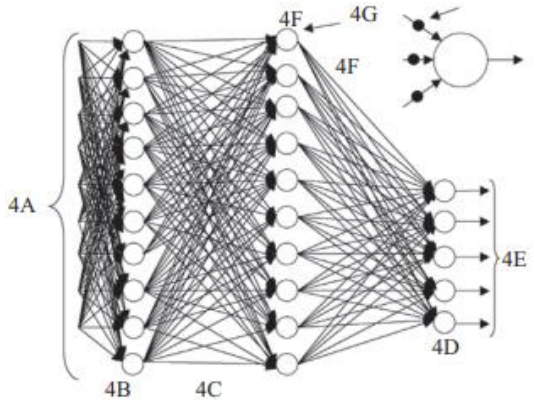


Figura 9-1. Estructura de una red neuronal multicapa.

Realizado por: Ponce Cruz, P. 2010.

1.8 Redes Neuronales Convolucionales

Una red neuronal convolucional, del inglés *convolutional neural network* (CNN) es un modelo de red neuronal que se inspira en el procesamiento de información en la neocorteza (la parte del cerebro que controla las funciones más complejas en los humanos), especialmente la corteza visual.

En 1959, los profesores David Hubel y Torsten Wiesel realizaron un trabajo experimental en el que registraron la actividad neuronal en la corteza visual del cerebro de un gato cuando veía una serie de imágenes. Para su sorpresa, encontraron que había neuronas que se activaban si las líneas que dibujaban tenían una orientación específica, a la que llamaron células simples. También notaron que otras neuronas se activaban si, los dibujos, además de presentar una orientación determinada, se desplazaban con una dirección específica, a las que llamaron células complejas. Observaron que el cerebro usa estas células para analizar información visual y las organiza en una estructura jerárquica, con células juntas analizando el área cercana del campo visual.

Uno de los primeros modelos desarrollados a partir de este trabajo fue el Neocognitron, que hoy se conoce como la inspiración para las redes neuronales convolucionales. Kunihiko Fukushima propuso un modelo de red neuronal llamado Neocognitron en 1980. Este modelo, basado en el trabajo de Hubel y Wiesel, es capaz de procesar imágenes a través de una serie de capas, encargadas de extraer características en orden secuencial, con el objetivo de imitar el extracción de características jerárquicas del cerebro.

En 1989, Yann LeCun y su equipo en AT&T Labs, propusieron la primera red neuronal convolucional, llamada LeNet-5, que se entrenó utilizando el algoritmo *backpropagation* (que no

existía cuando se introdujo el Neocognitron) para el reconocimiento de dígitos escritos a mano. (Pérez y Gegúndez 2021). El método de Yann LeCun se volvió tan importante que, a fines de la década de 1990, el 10% de los cheques que circulaban en los Estados Unidos eran controlados por redes convolucionales. Desde entonces hasta 2012, la gente estaba desarrollando aplicaciones utilizando CNN, pero no fue hasta 2012 que el campo de la visión por computadora se revolucionó cuando Geoff Hinton propuso un modelo que podía clasificar imágenes en 1,000 categorías diferentes. (Pusiol, 2014)

Las redes neuronales convolucionales son en si tipos especializados de redes neuronales que se utilizan para procesar datos que tienen una topología similar a una red. Por ejemplo, en los datos de series de tiempo, se pueden considerar como retículas unidimensionales muestreadas a intervalos de tiempo fijos, mientras que en los datos de imagen, se pueden considerar como retículas bidimensionales de píxeles. Las redes convolucionales son redes neuronales simples que utilizan operaciones matemáticas lineales de convolución en al menos una de sus capas, en lugar de la multiplicación de matrices habitual.

Las CNN también se denominan simplemente redes convolucionales, y el término *ConvNet* se ve a menudo en la literatura para referirse a ellas. En comparación con las ANN clásicas, como los perceptrones multicapa, las CNN funcionan mejor en imágenes del mundo real porque pueden distinguir patrones en datos multidimensionales, es decir, comprenden el hecho de que los píxeles cercanos entre sí en una imagen están más estrechamente relacionados que los píxeles que están más separados. Las CNN organizan sus neuronas en tres dimensiones: ancho, alto y profundidad. (Morocho Sande y Pulig Cesén, 2021)

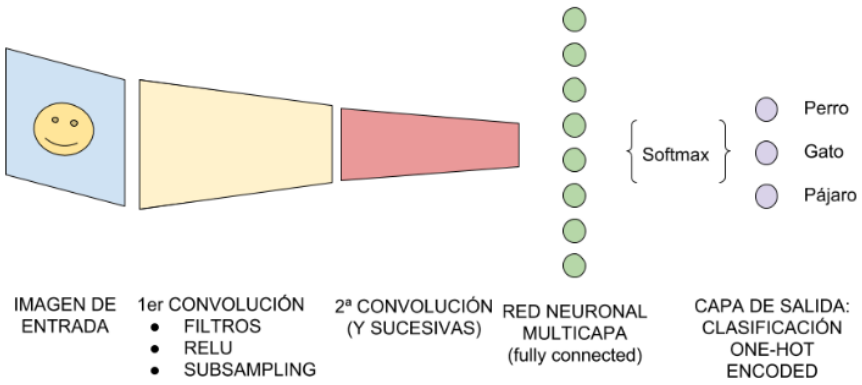


Figura 10-1. Estructura base de una red neuronal convolucional.

Realizado por: Na8, 2018.

Las redes convolucionales se utilizan generalmente para resolver una serie de problemas prácticos que requieren procesamiento de imágenes. Por ejemplo, si la cámara frontal de un automóvil autónomo captura una señal de tráfico, debe reconocer qué señal en particular es (clasificación de

imágenes). También puede detectar qué tipos de objetos se muestran en una imagen correspondiente a una escena e identificarlos en la misma imagen (detección de objetos). Como parte del sistema, las redes convolucionales también se pueden utilizar para generar descripciones textuales del contenido de la imagen. Esto permitiría indexar imágenes, buscar por contenido en bases de datos de imágenes o sintetizar señales de voz a partir de descripciones de texto para personas con discapacidad visual. (Berzal, 2018). En la Figura 10-1 se muestra la estructura básica de una CNN para identificación de objetos.

1.8.1 Estructura de una Red Neuronal Convolucional

Los componentes de una red neuronal convolucional constan de cinco capas diferentes, incluida una capa de entrada, una capa convolucional, una capa de agrupación o *pooling*, una capa de clasificador completamente conectada y una capa de salida.

1.8.1.1 Capa de entrada

La capa de entrada se puede definir como la primera capa, la puerta principal de la red neuronal convolucional. Utiliza imágenes en varios formatos como entrada, dependiendo de lo que se utilice para la tarea de estandarización.

Otra forma de verlo es que cuando una imagen se alimenta a la red, se trata como una matriz con valores de píxeles. Lo que ve la computadora depende del tamaño de la imagen. Como se mencionó anteriormente, está determinado por las dimensiones de altura, ancho y profundidad. Estos se determinan como valores de canal RGB en el rango de 0 a 255 píxeles, lo que se entiende como escala de grises. (Roman y Ruiz, 2021)

1.8.1.2 Capa convolucional

Estas capas ocultas están relacionadas con el aprendizaje de patrones locales en una pequeña ventana bidimensional de alto por ancho. Estas son las capas que se utilizan para detectar características visuales en la imagen, como líneas, bordes y aberraciones cromáticas. Estas capas nos permiten aprender los atributos de la imagen en puntos específicos y reconocerlos en cualquier parte de la imagen.

La capa convolucional puede aprender elementos diferentes y más complejos en función de lo que han aprendido las capas anteriores. Por ejemplo, una capa puede aprender los diferentes tipos de líneas que aparecen en una imagen, y la siguiente capa puede aprender los elementos de la imagen compuestos por las diferentes líneas aprendidas por la capa anterior. La capacidad de hacerlo permite que estas redes aprendan de manera efectiva conceptos visuales cada vez más complejos.

Las redes neuronales convolucionales se ejecutan en tres ejes: ancho, alto y otro eje que es la profundidad. Si la imagen de entrada es en escala de grises, la profundidad será 1. Si es una imagen en color RGB, la profundidad será de 3, uno por cada color (rojo, verde, azul).

No todas las neuronas de entrada estarán conectadas a todas las neuronas de esta capa convolucional. Esto se hace ingresando pequeñas regiones locales de píxeles de imagen que se almacenan en el espacio neuronal.

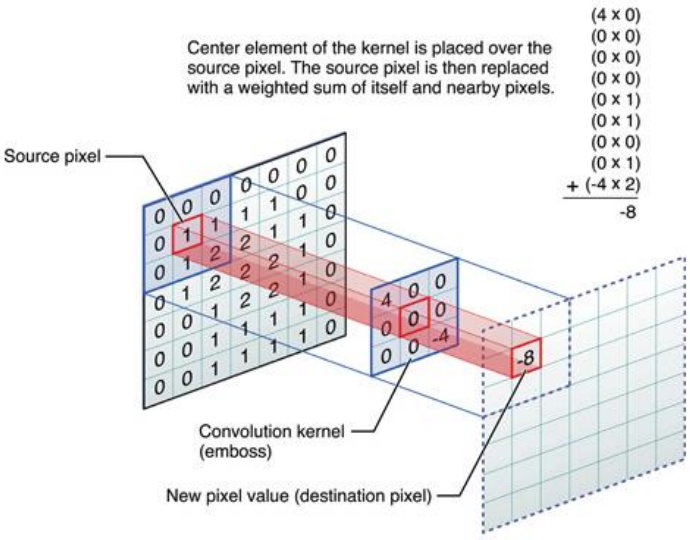


Figura 11-1. Proceso de convolución.

Realizado por: Apple, 2016.

Como puede verse en la Figura 11-1, esta matriz denominada *kernel*, atraviesa todas las neuronas de entrada. En cada posición de la matriz hay una neurona.

La matriz comienza con todas las neuronas en la parte superior izquierda y se mueve hacia la derecha. Cuando se completa una fila, continúa con la fila que está debajo. De esta forma, se recupera información de todas las neuronas y estas neuronas se conectan entre sí.

Así, podemos observar que el espacio neuronal definido por la salida es más pequeño que el espacio de entrada. Esto puede resultar en pérdida de información. Por tanto, para evitar la pérdida de neuronas, se aplica una técnica de llenado con ceros a la red neuronal. Esta capa convolucional aplica una serie de filtros, cada uno de los cuales detecta un atributo de la imagen. (Saez De La Pascua, 2019)

1.8.1.3 Capa de reducción o pooling

Su propósito es reducir el espacio dimensional sin perder información importante de la imagen, la cual ayuda a analizar la misma. Esto se debe a que después de pasar por la capa convolucional, la imagen podrá utilizar el clasificador elegido, pero se deben tener en cuenta estos inconvenientes

en cuanto a la complejidad computacional. Esto demuestra que la capa de *pooling* es muy importante. Hay diferentes tipos de capas de *pooling*, cada una con diferentes funciones. El *max pooling* se refiere básicamente a tomar el valor máximo de cada punto, independientemente de si el tamaño es 2-D o 3-D. El *average poolilng* en cambio, en lugar de obtener el valor máximo, obtiene el valor promedio y lo usa como resultado. (Roman y Ruiz, 2021)

En la Figura 12-1 se muestra el proceso de reducción de la imagen, donde se obtiene una matriz de dimensiones reducidas como resultado.

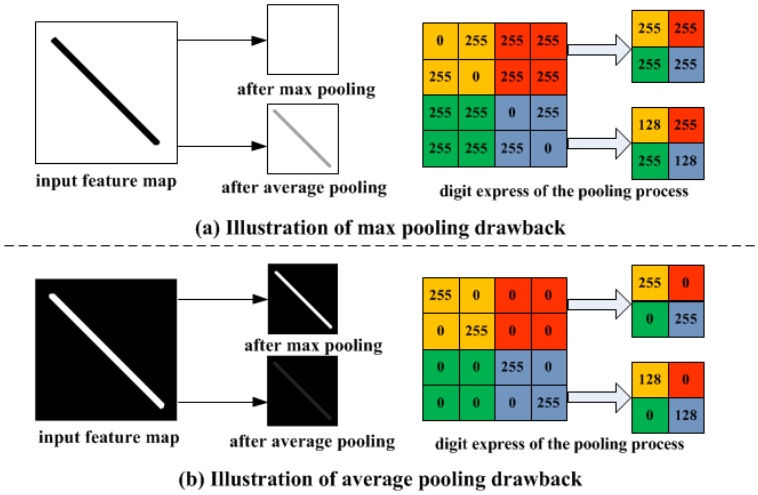


Figura 12-1. Tipos de *pooling*. (a) *Max pooling*. (b) *Average pooling.*

Realizado por: Yu, D.; et al. 2014.

1.8.1.4 Capa clasificadora totalmente conectada

Este tipo de capa es exactamente igual que cualquier capa en una red neuronal artificial tradicional, pero con una arquitectura completamente conectada. Cada neurona de esta capa está conectada a cada neurona de la capa anterior. Más precisamente, está conectado a todos los mapas de características de la capa anterior.

A diferencia de lo que hemos visto hasta ahora en redes neuronales convolucionales, este tipo de capa no usa la propiedad de conexión local, ya que una capa completamente conectada está conectada a cada entrada. La capa convolucional está conectada a la región local de la entrada y muchas de las neuronas de la capa convolucional comparten parámetros.

En este tipo de capa, el único parámetro configurable es el número de neuronas que la componen. Estas capas conectan las neuronas K a todas las entradas de cada neurona K. Su salida será un único vector $1 * 1 * K$ que contiene las activaciones calculadas. Este hecho muestra que no hay capas convolucionales después de esta capa, desde la entrada 3D hasta un solo vector de salida 1D.

La función principal de la capa totalmente conectada es realizar una especie de agrupamiento sobre la información adquirida hasta el momento, y esta información se utilizará en cálculos posteriores para lograr la clasificación final.

Normalmente, en las redes neuronales convolucionales, se utilizan varias capas completamente conectadas en serie, y la última de ellas tiene un parámetro K, que es el número de clases en el conjunto de datos. El valor final de K se envía a la capa de salida, donde se clasifica mediante una función de probabilidad específica.

Hay otros parámetros que se pueden configurar, como pesos y valores de desviación, pero estos no son estrictamente necesarios y se pueden utilizar valores predeterminados. (Bonilla Carrión, 2020)

1.8.1.5 Capa de salida

Después de conocer la función de cada capa, la arquitectura de CNN entra en la fase de clasificación. La penúltima capa está completamente conectada y genera un vector de dimensiones K. Aquí, K es el número de clases que la red puede predecir. Este vector contiene las probabilidades de que se clasifique cada categoría de imágenes. La última capa de la arquitectura de CNN utiliza una capa de clasificación como *softmax* para proporcionar la salida de clasificación. (Matlab, 2021)

1.8.2 Tipos de clasificación con redes neuronales convolucionales

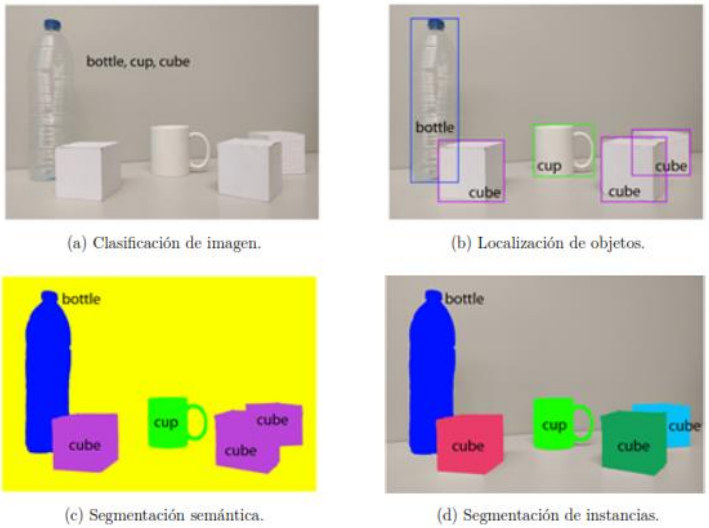
Existen muchos algoritmos diferentes para clasificar objetos en una imagen, cada uno con una utilidad y complejidad variable según la tarea que resuelven. Estos algoritmos se muestran en la Figura 13-1.

A continuación, analizaremos brevemente los diferentes algoritmos en orden ascendente de complejidad.

- La clasificación de imágenes (Figura 13-1a) implica obtener la categoría o etiqueta de un objeto en una imagen, sin especificar su ubicación, o si hay más de uno, es decir, en forma binaria. Normalmente, se utiliza una imagen centrada en un único objeto grande, pero no es necesario.
- La localización de objetos (Figura 13-1b) implica clasificar la imagen de manera similar al algoritmo de "clasificación de imágenes", pero en este caso agregando un rectángulo alrededor del objeto a localizar, llamado cuadro delimitador o *bounding box*.
- La segmentación semántica (Figura 13-1c) implica la clasificación de todos los píxeles de una imagen, donde cada píxel está asociado con una etiqueta correspondiente. De esta forma, se puede recuperar un conjunto de píxeles clasificados con la misma etiqueta. Con este

algoritmo, no es posible conocer el número de instancias de objetos, solo el número de píxeles marcados o clasificados con una etiqueta en particular.

– La segmentación de instancias (Figura 13-1d) es una variante del algoritmo de "segmentación semántica", que puede clasificar solo los píxeles que pertenecen a un objeto. Este algoritmo nos permite obtener el número de instancias de un objeto en una imagen. Puede verse como una evolución de la "detección de objetos" debido a su uso similar y mayor precisión, ya que no tiene por qué limitarse al posicionamiento a través de cajas rectangulares. (Díaz Pedrayes, 2021)



(a) Clasificación de imagen. (b) Localización de objetos.

(c) Segmentación semántica. (d) Segmentación de instancias.

Figura 13-1. Ejemplo de los diferentes tipos de clasificación de una CNN.

Realizado por: Garcia Garcia, A.; et al. 2017.

1.9 Segmentación semántica

La segmentación semántica se considera una tarea avanzada que conduce a un mejor conocimiento de la escena, y consiste en asignar una etiqueta semántica a cada píxel de la imagen con el fin de identificar todos los elementos que componen la escena. Sin embargo, para realizar esta difícil tarea, es necesario considerar que la segmentación semántica no es una actividad aislada en el campo de la visión por computador; los distintos tipos de elementos que lo componen deben ser previamente clasificados y localizados. La Figura 14-1 muestra un ejemplo de un algoritmo de segmentación semántica. Cada píxel está etiquetado con una de las categorías definidas previamente.

En base a lo anterior, se han propuesto dos aspectos que pueden ayudar a mejorar la calidad de los resultados de la segmentación semántica. El primero es diseñar representaciones de características que puedan distinguir entre diferentes tipos de objetos. El segundo es utilizar información contextual para garantizar la coherencia entre las etiquetas de píxeles, es decir, etiquetar cada píxel de forma coherente.

Figura 14-1. Segmentación semántica de una imagen.

Realizado por: Orozco, O.; et al. 2019.

La segmentación semántica aporta mucha información para entender la escena, pero aún queda un camino complejo para definir las diferentes relaciones entre las clases en la imagen para facilitar la comprensión de la escena. Aunque varios autores han propuesto modelos que han logrado buenos resultados, se considera un problema abierto para aplicaciones del mundo real debido a los diversos obstáculos que pueden surgir y que afectan la calidad de los resultados de la segmentación semántica. (Prieto Ordaz, Ramírez Alonso y Maloof Flores, 2019)

1.9.1 Métodos de segmentación semántica

La segmentación de imágenes es un área de investigación importante en la visión por computadora y sus métodos han evolucionado continuamente a lo largo de los años. Como resultado, a lo largo de la historia de la segmentación, podemos encontrar diferentes métodos y diferentes tipos de aplicaciones. Ya sea la naturaleza o lo real, un paisaje o un comportamiento humano, una escena de tráfico o de carácter médico. (Monasterio Expósito, 2018)

Con base en la literatura disponible actualmente sobre diferentes métodos de segmentación y cómo se realiza cada una de ellas, existen varias clasificaciones. En este caso, con base en las teorías desarrolladas, se enumeran los siguientes tipos de métodos de segmentación:

- Métodos basados en la intensidad
- Métodos basados en regiones o similitudes
- Métodos basados en discontinuidad
- Métodos basados en agrupación en clústeres

1.9.1.1 Método basado en la intensidad

Es uno de los métodos de segmentación más populares, antiguos y relativamente simples en el procesamiento de imágenes. El proceso de umbralización se basa en un histograma. El conjunto de valores de gris de todos los píxeles, agrupados cuantificando el número de píxeles con el mismo

valor de gris, se denomina histograma. Los picos y valles de estos conjuntos de valores, llamados umbrales, se utilizan para determinar los grupos o clústeres en los que se pueden clasificar los píxeles. Los píxeles con un valor gris entre estos valores se asocian con el grupo correspondiente. En algunos casos, se realiza un proceso recursivo para volver a separar cada grupo en subgrupos y para distinguir los picos y valles de las subregiones. Esto significa que, se puede basar en características globales de la imagen (histograma de toda la imagen) o características locales (si se seleccionan umbrales para ciertas regiones o vecindarios). En el Gráfico 1-1-1 se muestra un ejemplo de histograma.

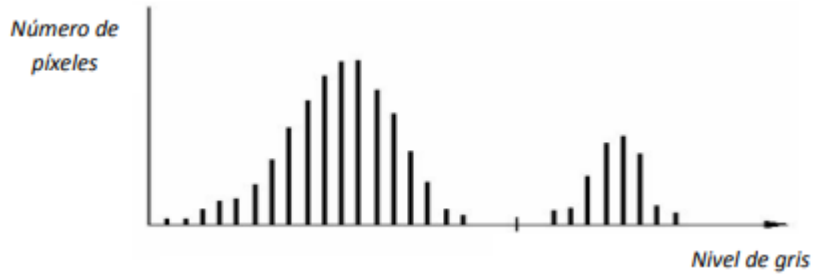


Gráfico 1-1. Ejemplo de un histograma de una imagen en escala de grises.
Realizado por: Pérez, A. 2016.

La forma más sencilla de hacerlo es determinar específicamente el nivel de umbral, dividir los píxeles en dos grupos, determinar si pertenecen a esta categoría y convertir la imagen inicial al nivel de gris de una imagen binaria. Hay varias modificaciones. Podemos señalar dos o más umbrales (determinados por los picos o valles del histograma) para dividir la imagen en varios niveles. Algunos de los métodos son adaptativos. Según el análisis del algoritmo del histograma local cerca de un píxel, los umbrales pueden ser diferentes en diferentes áreas de la imagen y se puede determinar el tamaño de esta vecindad. Un método muy conocido es el método de Otsu, que propone un umbral óptimo para maximizar la separación entre clases. (Pérez, 2016)

1.9.1.2 Métodos basados en las regiones

Otro método tradicional de segmentación de imágenes es el método basado en regiones. En este método, la imagen original se divide en varias regiones. Los píxeles de estas regiones tienen propiedades específicas.

Este proceso de segmentación se puede realizar utilizando dos métodos diferentes, generalmente denominados *Growing Region* y *Split and Merge Region*. El primer método, *Growing Region*, comienza con una región semilla de un tamaño determinado y crece de forma iterativa utilizando la similitud entre los píxeles de la región y los píxeles vecinos que no pertenecen a la región. Se pueden tener en cuenta aspectos como la intensidad, la textura y el color al comprobar las

propiedades de cada píxel vecino. Por tanto, después de incluir todos los píxeles vecinos que proporcionan uniformidad a la región, los píxeles vecinos bajo los que crece la semilla inicial comenzarán a tener propiedades completamente diferentes, dividiendo así una región separada de la región inicial. La Figura 15-1 a continuación, muestra un ejemplo de crecimiento de semillas y cómo el número de píxeles en la región aumenta gradualmente hasta que se encuentra un objeto, en este caso un pétalo.

Figura 15-1. Funcionamiento del método *Growing Region*.

Realizado por: Palomino, N.; Concha, U. 2009.

Por otra parte, el método *Split and Merge Region* se basa en una idea ligeramente diferente. En este caso, la imagen original se segmenta previamente en diferentes regiones, cuyo número se determina al comienzo de la segmentación. Luego, cada una de estas regiones se analiza por separado para que puedan segmentarse o combinarse más tarde. Este proceso se basa en la inspección de los píxeles en una región en particular. Si hay una ligera diferencia en los atributos entre ellos, la región se divide en subregiones más pequeñas para que cada subregión cumpla con los criterios de uniformidad. La situación puede invertirse, donde varias regiones se combinan en una región suficientemente uniforme.

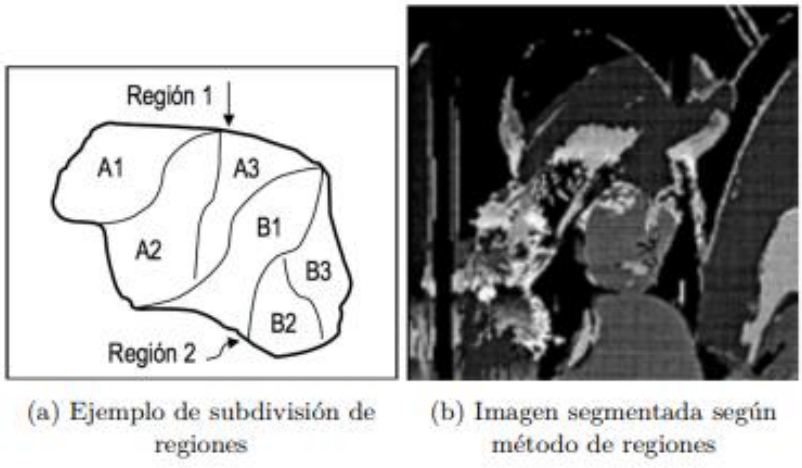


Figura 16-1. Funcionamiento del método *Split and Merge Region*.

Realizado por: Palomino, N.; Concha, U. 2009.

El objetivo explícito de estos dos métodos es garantizar la segmentación en la imagen, donde los diferentes objetos de la imagen están completamente separados, así como la expansión de la región. La Figura 16-1a muestra un ejemplo del método *Split and Merge Region*: las dos regiones iniciales se han subdividido y el número total de regiones se ha incrementado. La Figura 16-1b muestra el resultado de segmentar una imagen usando el método de las regiones. (Monasterio Expósito, 2018)

1.9.1.3 Métodos basados en discontinuidad

Otro método de segmentación muy relevante en este campo se basa en el método de discontinuidad. Los métodos de discontinuidad segmentan una imagen según los cambios bruscos en el nivel de gris que pueden aparecer en la imagen. Estas discontinuidades generalmente se pueden detectar en función de la ubicación de puntos aislados, líneas o la detección de bordes y contornos en la propia imagen. Este último es el método más utilizado para la segmentación basada en discontinuidades. Es por eso que el uso de detección de bordes o *Edge Detection* es tan común. Esto se debe a que la escala de grises sirve para dividir claramente dos regiones adyacentes que son claramente diferentes. Por lo tanto, los bordes se pueden clasificar de acuerdo con su ancho, el ángulo de cambio de pendiente y las coordenadas del punto medio, por lo que los bordes se pueden clasificar en tipos lineales, escalonados, de rampa y de techo. Después de discutir cómo clasificar los bordes, es importante saber cómo reconocer estas discontinuidades. Para hacer esto, usaremos la primera derivada, el operador de gradiente, y la segunda derivada donde se puede encontrar el operador laplaciano. (Monasterio Expósito, 2018)

1.9.1.4 Método basado en agrupamiento

Los métodos de agrupación, también conocidos como algoritmos de *clustering*, se basan en la combinación de diferentes subconjuntos de datos de entrada, denominada agrupación. Por lo tanto, para utilizar este tipo de algoritmo en la segmentación de imágenes, los píxeles pertenecientes a la imagen de entrada deben clasificarse en diferentes clústeres y las características de los elementos que forman cada clúster deben dar como resultado patrones o atributos similares, creándolos simultáneamente con otros clústeres.

Los métodos de agrupación se caracterizan por dividir las agrupaciones según la distancia u otros criterios, utilizando funciones de equidistancia euclidiana o según la similitud misma. Por tanto, los clústeres pertenecen a la categoría de algoritmos de segmentación no supervisados, donde los datos de entrada son una variable aleatoria, a diferencia de los algoritmos supervisados que se basan en conocimientos previos.

Por lo tanto, los métodos basados en clústeres se pueden clasificar en dos categorías. En la primera categoría, se realiza la agrupación jerárquica. En esta categoría, necesitamos realizar la partición

de clústeres y la posible integración entre clústeres. En el segundo grupo, se denomina agrupación no jerárquica. Esto significa que el número de conglomerados necesarios es fijo y la decisión de pertenecer a cada conglomerado se toma mediante un criterio de distancia.

En cuanto al uso de este algoritmo basado en clústeres, es importante mencionar que es muy utilizado en minería de datos, bioinformática y procesamiento de imágenes. Este campo es relevante para la investigación realizada en este trabajo. (Monasterio Expósito, 2018)

1.10 Procesamiento Digital de Imágenes

El conjunto de técnicas y procesos que utilizan las computadoras como herramienta principal para descubrir o resaltar información contenida en una imagen se denomina procesamiento digital de imágenes (PID). Actualmente, el PID es un área de investigación muy específica en el campo de la informática y está estrechamente relacionada con el procesamiento de señales digitales. Esta relación se debe al hecho de que el PDI es esencialmente una forma muy especial de procesamiento de señales digitales bidimensionales o tridimensionales. (Torres, 1996). El interés por la investigación de PDI se basa en dos áreas de aplicación principales:

- Mejorar la calidad de la información contenida en una imagen y hacer que esta información esté disponible para la interpretación humana.
- Procesamiento de los datos contenidos en una escena con una percepción autónoma.

1.10.1 Operaciones morfológicas

Las transformaciones morfológicas son responsables de cambiar la forma y estructura de un objeto. Estas herramientas matemáticas le permiten hacerse una idea de los componentes de la forma y estructura de los objetos que componen la imagen. Además, estas formas pueden modificarse para separar objetos entre sí, obtener contornos importantes, descomponer formas complejas en formas más simples, obtener contornos en entornos ruidosos y reconstruir elementos distorsionados. Normalmente, este conjunto de transformaciones se realiza en imágenes previamente binarizadas (en blanco y negro), pero algunas investigaciones han aplicado estas técnicas a imágenes en escala de grises. Las transformaciones a procesar corresponden a la morfología matemática, y su lenguaje de expresión es la teoría de conjuntos. (Marcos et al., 2006). A continuación veremos las operaciones morfológicas más usadas en PDI.

1.10.1.1 Dilatación

Dados dos conjuntos A y B, el proceso de dilatación, representado por $A \oplus B$, es definida como:

$$A \oplus B = \{c \in E^N | c = a + b; para\ todo\ a\ \in A\ y\ todo\ b \in B$$

En otras palabras, dada una máscara B (compuesta de unos y ceros), la dilatación de A por B es un conjunto de todos los cambios de x de modo que B y A se superponen al menos con un elemento distinto de cero. Por ejemplo, se nos da una máscara B formada por una matriz:

0	1	0
1	1	1
0	1	0

Figura 17-1. Matriz máscara B.

Realizado por: Marcos, A.; et al. 2006.

Y una matriz A:

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Figura 18-1. Matriz A.

Realizado por: Marcos, A.; et al. 2006.

El proceso de dilatación $A \oplus B$ implica, para cada punto $A\ (i, j)$, superponer la máscara B de A (el punto central de B es (i, j)) y establecer el punto $A\ (i, j)$ en 1 si cualquiera de las B corresponde a A. En otras palabras, la dilatación implica mover los elementos estructurantes (B) alrededor de la imagen. Si un elemento de la matriz (llamado píxel adyacente en la máscara) coincide con un píxel del entorno, el píxel central de A se establece en 1, el resultado de la operación se muestra en la Figura 19-1. (Marcos et al., 2006). En la Figura 20-1 se muestra un ejemplo de dilatación de una imagen binarizada.

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Figura 19-1. Matriz resultante de la dilatación entre A y B.

Realizado por: Marcos, A.; et al. 2006.

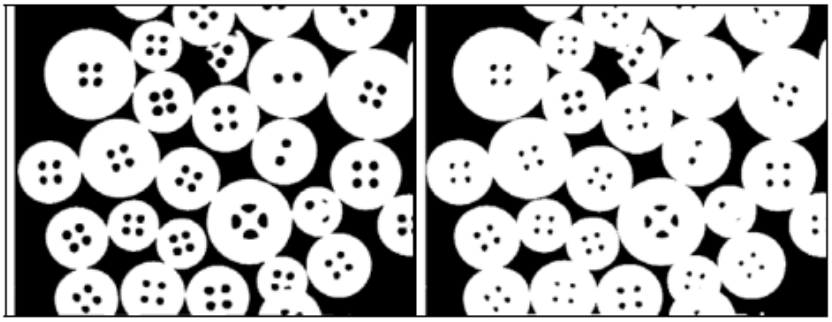


Figura 20-1. Imagen cualquiera binarizada (izquierda) y su dilatación (derecha).

Realizado por: Marcos, A.; et al. 2006.

1.10.1.2 Erosión

Dados dos conjuntos A y B, el proceso de erosión, representado por $A \ominus B$, es definido como:

$$A \ominus B = \{x \in E^N | x = x + b \in A; para\ todo\ b \in B$$

La erosión es una doble función de la dilatación (pero no al revés. Es decir, si se realiza primero la erosión y luego la dilatación, la imagen no mantendrá su aspecto original). Si la imagen A es erosionada por B, entonces para cada punto x, después de ser transformado por x, B se incluirá en A. En otras palabras, la erosión pone a cero todos los píxeles de la imagen que no contienen completamente elementos estructurales en el entorno. A medida que la dilatación expande los bordes y los contornos de un objeto, la erosión reduce los contornos del objeto. Se utiliza para separar objetos que están conectados por pequeñas partes del contorno. Por ejemplo, haciendo uso de la máscara del caso anterior:

0	1	0
1	1	1
0	1	0

Figura 21-1. Matriz máscara B.

Realizado por: Marcos, A.; et al. 2006.

y una matriz A:

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

Figura 22-1. Matriz A.

Realizado por: Marcos, A.; et al. 2006.

En el proceso de erosión $A \ominus B$, para cada punto $A\ (i, j)$, se superpone la máscara B sobre A (el punto central de B es (i, j)), y los puntos de la máscara B se emparejan haciendo clic en la imagen y estableciéndose a 1. De lo contrario, se pondrá a cero. En otras palabras, la dilatación implica mover el elemento estructural (B) a través de la imagen. Si un elemento en la matriz (llamado píxel vecino de la máscara) no coincide con un píxel en particular en el entorno, se establecerá en 0 el píxel centrado en A, se establece en 1 si todo coincide con los elementos de la máscara, el resultado de la erosión se muestra en la Figura 23-1. (Marcos et al., 2006). La Figura 24-1 muestra el proceso de erosión en una imagen binaria.

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Figura 23-1. Matriz resultante de la erosión entre A y B.

Realizado por: Marcos, A.; et al. 2006.

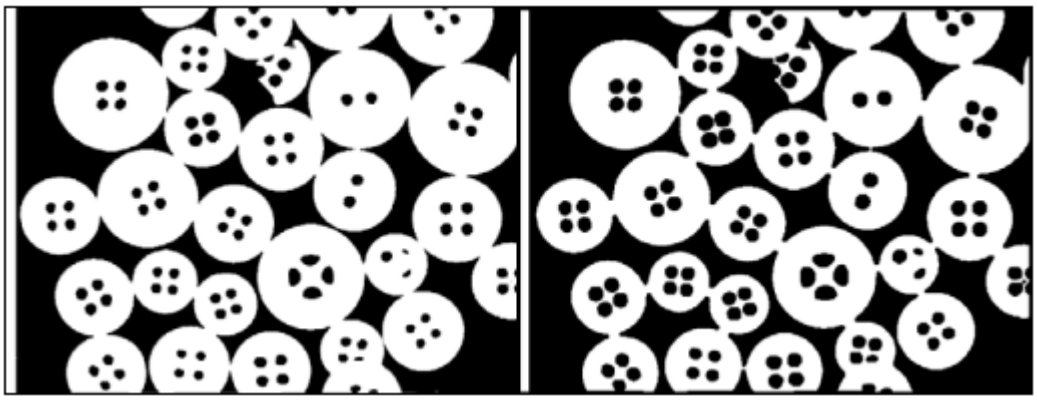


Figura 24-1. Imagen binarizada (izquierda) y su erosionado (derecha).

Realizado por: Marcos, A.; et al. 2006.

1.10.1.3 Apertura

La apertura implica erosionar una imagen y luego expandirla. A primera vista parecen iguales, pero no lo son, porque la expansión no es lo opuesto a la erosión. La Figura 25-1 muestra un ejemplo de apertura de una imagen binarizada, en el cual se puede apreciar que este proceso nos permite eliminar el 'ruido' de la imagen.

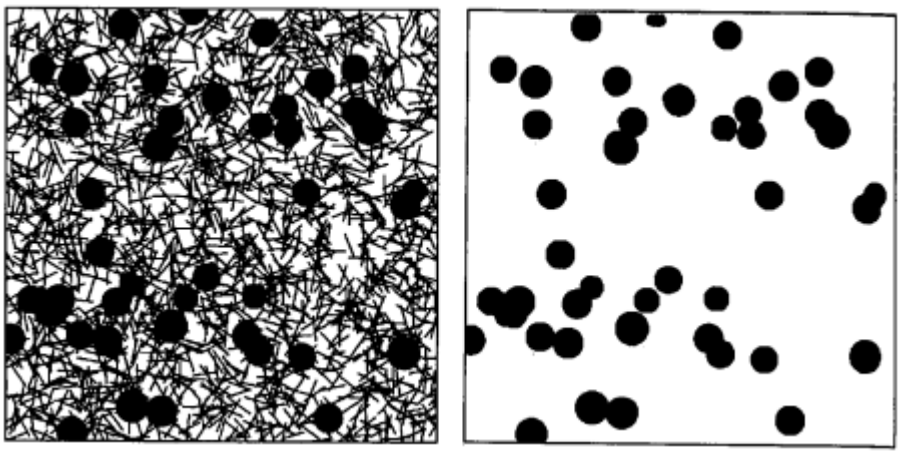


Figura 25-1. Imagen con ruido (izquierda) filtrada mediante proceso de apertura, aplicando una máscara en forma de disco (derecha).
Realizado por: Marcos, A.; et al. 2006.

La apertura se usa ampliamente para:

- Segmentar un objeto, separando algunas formas de otras.
- Divide un objeto en elementos más simples.
- Extrae formas específicas en entornos ruidosos.
- Elimina proyecciones estrechas.
- Separe los objetos que no estén demasiado juntos.
- Agrande los agujeros dentro de los objetos.

Las aberturas pueden formarse por una variedad de erosión y expansión, con resultados muy variables. (Marcos et al., 2006)

1.10.1.4 Cierre

El efecto de cierre es exactamente lo opuesto al efecto de apertura, primero expansión y luego erosión. Las combinaciones de operaciones de apertura y cierre se pueden utilizar para múltiples operaciones, como filtrado, segmentación, etc. modificando el tipo y tamaño de la máscara. En la Figura 26-1, puede ver que simplemente abriendo y cerrando la figura, puede dividirla en su base, tronco y partes horizontales.

A partir de una imagen con ruido impulsivo (Figura 26-1a), se perfora un orificio en forma de disco de radio 1 (Figura 26-1b). Cerrado con un disco de radio 4, se rellena el espacio dejado por el paso anterior (Figura 26-1c). El borde vertical se eliminará mediante una abertura de radio 3 (Figura 26-1d). Se usa una máscara rectangular para extraer el sustrato (Figura 26-1e). Reste la Figura 26-1e de la Figura 26-1d para determinar los residuos (Figura 26-1f). A través de la máscara formada por la estructura vertical, se obtiene el tronco del árbol (Figura 26-1g).

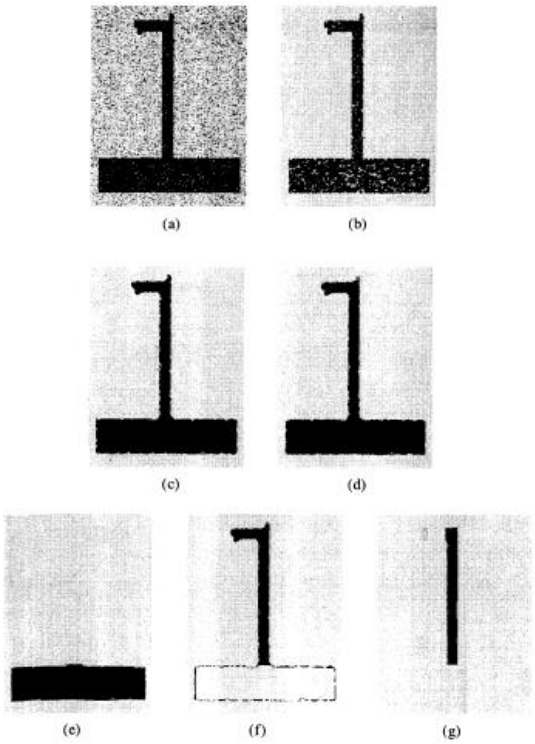


Figura 26-1. Los componentes gráficos se extraen abriendo y cerrando.

Realizado por: Marcos, A.; et al. 2006.

El uso de apertura y cierre es enormemente costoso en términos de tiempo de ejecución del equipo, pero la simplicidad del mecanismo permite implementarlo directamente en hardware, simplificando la tarea de segmentación en tiempo real y reconocimiento de objetos simples. (Marcos et al., 2006)

1.11 Matlab

MATLAB es un lenguaje de alto rendimiento para informática técnica. Integra la computación, la visualización y la programación en un entorno fácil de usar. Los problemas y las soluciones están representados por símbolos matemáticos familiares. Las aplicaciones típicas incluyen:

- Matemáticas y computación
- Desarrollo de algoritmos
- Modelado, simulación y creación de prototipos
- Análisis, investigación y visualización de datos
- Gráficos científicos y de ingeniería
- Desarrollo de aplicaciones, incluido el establecimiento de interfaces gráficas de usuario

MATLAB es un sistema interactivo y sus elementos de datos básicos son matrices que no necesitan ser dimensionadas. Esto le permite resolver muchos problemas técnicos de cálculo, especialmente aquellos que involucran ecuaciones matriciales y vectoriales. El tiempo requerido

es una fracción de lo que se necesitaría para crear un programa en un lenguaje escalar no interactivo como C o Fortran.

El nombre MATLAB significa *Matrix Laboratory*. El propósito original de crear MATLAB era facilitar a las personas el uso del software matricial desarrollado por los proyectos LINPACK y EISPACK, que juntos representan el último nivel de software de computación matricial.

Después de muchos años de desarrollo, MATLAB ha sido respaldado por un gran número de usuarios. En un entorno universitario, es una herramienta de enseñanza estándar para cursos introductorios y avanzados en matemáticas, ingeniería y ciencias. En la industria, MATLAB es la herramienta preferida para la investigación, el desarrollo y el análisis productivos.

MATLAB se caracteriza por un conjunto de soluciones de aplicación específicas, denominado caja de herramientas. Para la mayoría de los usuarios de MATLAB, las cajas de herramientas son muy importantes porque les permiten aprender y aplicar métodos especializados. Toolbox es una colección completa de funciones de MATLAB (archivos M) que pueden extender el entorno de MATLAB para resolver tipos específicos de problemas. Las áreas donde se puede utilizar Toolbox incluyen procesamiento de señales, sistemas de control, redes neuronales, lógica difusa, ondas, simulación y muchas más. (Matlab, 2021)

CAPITULO II

2. METODOLOGÍA

En esta sección, explicaremos cómo extraer características de una imagen haciendo uso de CNN y PDI. El objetivo es asociar una categoría, etiqueta o categorías específicas a cada píxel presente en la imagen. Por lo tanto, se busca determinar si la imagen de satélite podría contener recursos hídricos y determinar su área. El método desarrollado incluye los siguientes pasos:

- Recolección y selección de los datos: Se utilizará un conjunto de imágenes satelitales de algunos recursos hídricos, en este caso, lagos y lagunas. La segmentación semántica de las imágenes se realizará en dos clases: agua y tierra firme.
- Entrenamiento del modelo: Para cumplir el objetivo de realizar una segmentación semántica de las imágenes, se entrenará un modelo de red neuronal basada en la arquitectura de red convolucional Resnet-18.
- Evaluación del modelo: La evaluación se realizará principalmente en función de las imágenes de prueba y su contraparte previamente etiquetada.

En el Gráfico 1-2 se muestra el flujo de trabajo que será tratado en este capítulo.

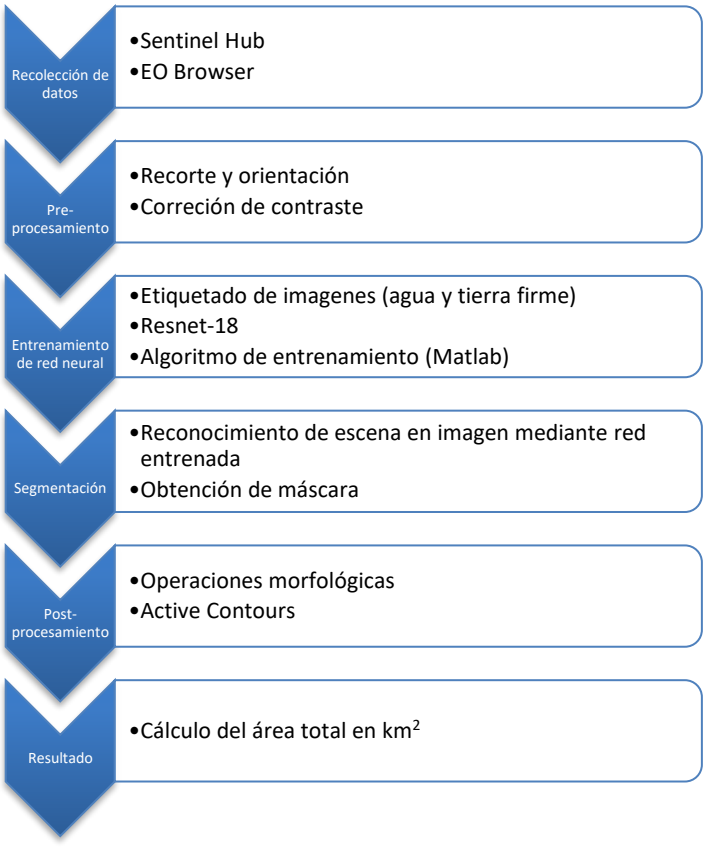


Gráfico 1-2. Flujo de trabajo para el cálculo de área de recursos hídricos.

Realizado por: Núñez, Christian, 2021.

2.1 Recolección de datos

En esta etapa de recopilación de imágenes de prueba, buscamos a través de diferentes fuentes para adquirir imágenes que sean adecuadas para nuestro propósito. Con fines más educativos y para poder realizar nuestro trabajo de acuerdo con aplicaciones específicas, seleccionamos estratégicamente un conjunto de imágenes satelitales. Luego, estas imágenes se organizan e integran para formar un conjunto de datos de entrada como punto de partida.

2.1.1 Sentinel Hub

Sentinel Hub proporciona un acceso sin precedentes a los datos de observación de la Tierra con un enfoque en los satélites Sentinel y soporte adicional para fuentes como *Landsat* y *Planet*. Al facilitar la tecnología avanzada en la nube de *Amazon Web Services* y los métodos innovadores para procesar y distribuir datos de manera eficiente en cuestión de segundos sin un procesamiento previo intensivo, ofrece una manera fácil de usar y rentable de explotar los datos en cualquier aplicación GIS o integrarlo en aplicaciones web. Sentinel Hub elimina así la gran molestia de descargar, archivar y procesar *petabytes* de datos y simplemente hace que el archivo global completo esté disponible a través de servicios web. Por tanto, los desarrolladores de aplicaciones pueden centrarse en servicios de valor añadido y aplicaciones para el usuario final en lugar de ocuparse de la complejidad de los datos de teledetección. De esta manera, Sentinel Hub aumentará la aceptación del programa Copernicus. (Copernicus Masters, 2016)

Sentinel Hub es un motor de procesamiento de datos por satélite. Proporciona acceso a la visualización y análisis de imágenes de observación de la Tierra, especialmente del Sentinel.

Sentinel Hub ofrece dos modos para explorar estas imágenes, a saber, Sentinel Playground y EO Browser. El primero proporciona un visor web simple para ver imágenes actualizadas diariamente de Sentinel 1, Sentinel 2, Landsat 8 y otros. Con EO Browser, se puede buscar y comparar imágenes de resolución completa de todas las fuentes de datos disponibles.

Se empleará EO Browser para ver y descargar imágenes. También nos permite personalizar la visualización de sus imágenes. Esto se puede hacer combinando diferentes bandas o usando scripts. Otra opción útil que se implementó este año es “fijar y comparar”. Esto permite a los usuarios seleccionar imágenes tomadas en diferentes momentos y verlas y compararlas simultáneamente.

Actualmente, Sentinel Hub solo ofrece el producto Sentinel-1 Level-1 GRD ofrecido por la ESA. Para los productos Sentinel-2, se ofrecen LC1 sin corrección atmosférica y L2A corregido para efectos atmosféricos, con una selección de porcentajes de cobertura de nubes. (Solá, 2021)

2.1.1.1 EO Browser

EO Browser es una herramienta web de código abierto para navegar, visualizar y analizar imágenes satelitales disponible en varias plataformas:

- Sentniel-1,
- Sentinel-2,
- Sentinel-3,
- Sentinel-5P

La aplicación hace posible que cualquier persona con Internet y un navegador web averigüe qué datos están disponibles en una ubicación específica, los visualice instantáneamente a resolución completa, realice varios análisis (por ejemplo, NDVI, NDWI, EVI y otros algoritmos, muchos de los cuales son disponible en GitHub), comparar datos de varias adquisiciones, crear lapsos de tiempo, exportar datos para su posterior análisis en herramientas GIS de escritorio, etc.

La facilidad de uso y la gran cantidad de datos disponibles lo convierten en una excelente herramienta para que los no expertos monitoreen el área en su vecindario o en el otro lado del mundo. EO Browser también se usa a menudo en educación, ya que permite a los estudiantes explorar el poder de los datos de observación de la Tierra. (Sinergise Laboratory for geographical information systems, 2021). La Figura 1-2 muestra la interfaz de EO Browser donde se aprecian las herramientas básicas que posee.

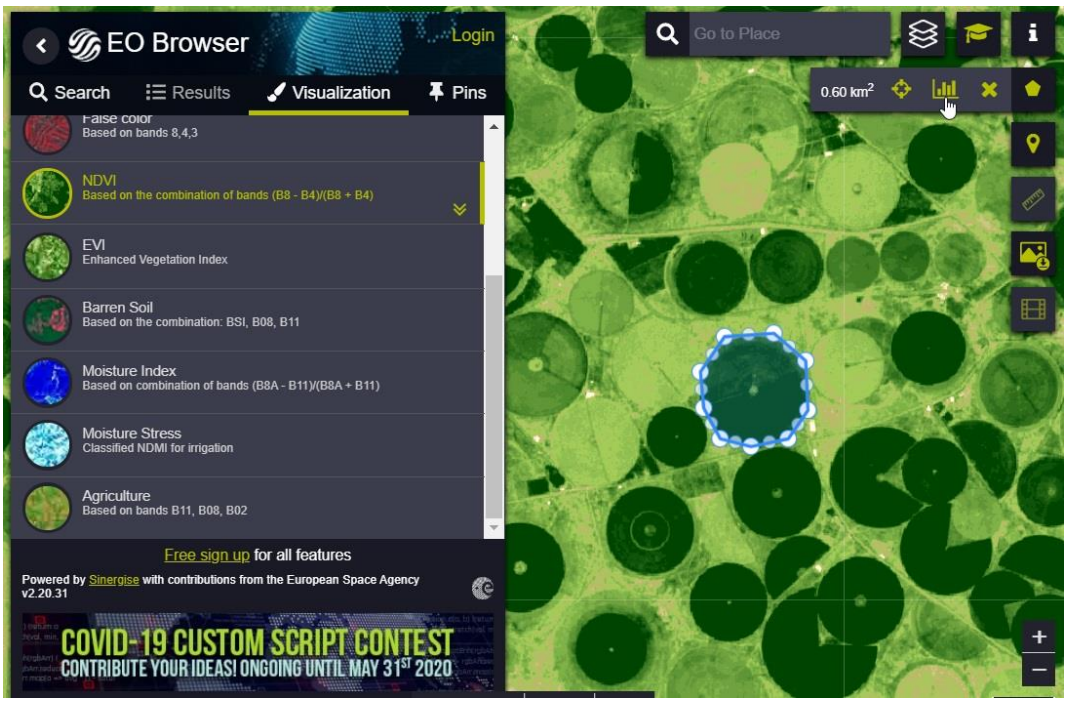


Figura 1-2. EO Browser - Espacio de trabajo.

Realizado por: Sinergise Laboratory for geographical information systems, 2021.

Las imágenes de satélite en EO Browser se pueden visualizar según la configuración deseada por el usuario. Existen varias visualizaciones con leyendas y descripciones preparadas para el usuario, como color verdadero, color falso, NDVI, EVI, etc. Al elegir el modo personalizado, es posible elegir cualquier combinación de bandas y hacer una composición simplemente arrastrando y soltando las bandas en los canales RGB.

Cualquier imagen que vea en la pantalla de EO Browser se puede descargar en varios formatos de archivo. Puede descargar una imagen rápida PNG o JPG, seleccionando si desea mantener el logotipo, la leyenda o los subtítulos, puede descargar imágenes de alta resolución o datos TIFF para un análisis posterior, todo gratis. (Sinergise Laboratory for geographical information systems, 2021)

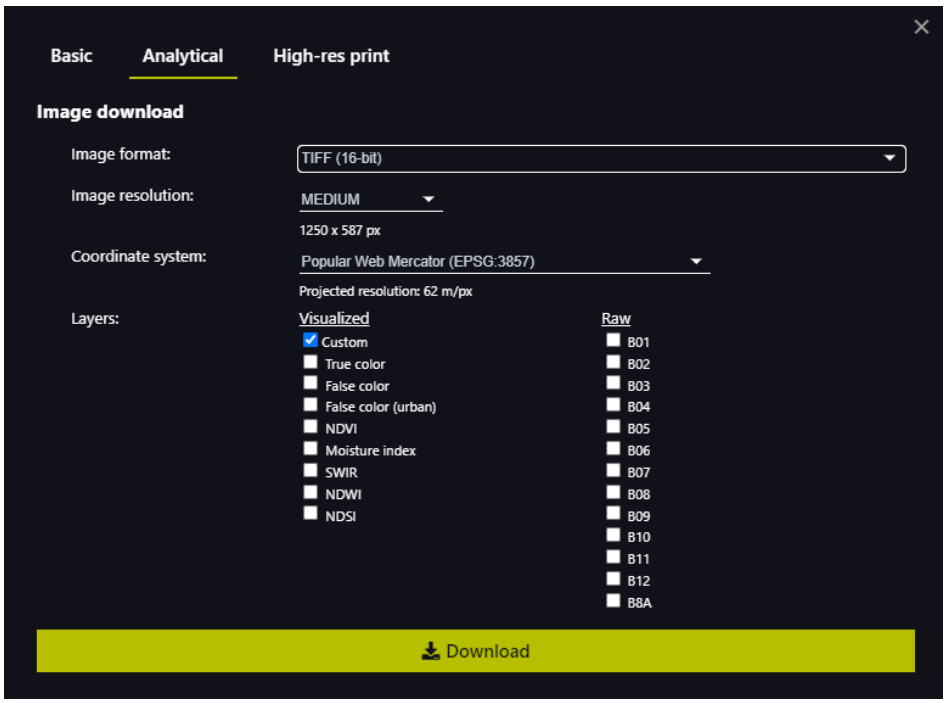


Figura 2-2. Ventana de descarga de imágenes de EO Browser.

Realizado por: Sinergise Laboratory for geographical information systems, 2021.

Como se muestra en la Figura 2-2, EO Browser nos permite descargar las imágenes pudiendo elegir el formato, la resolución, y las diferentes capas o bandas. Si se eligen más de una capa o banda, las imágenes se descargarán en un archivo comprimido, caso contrario, se descargará la imagen en el formato que hayamos elegido. En la Figura 3-2 se muestra un ejemplo de un conjunto de imágenes descargadas del mismo recurso pero en diferentes capas.

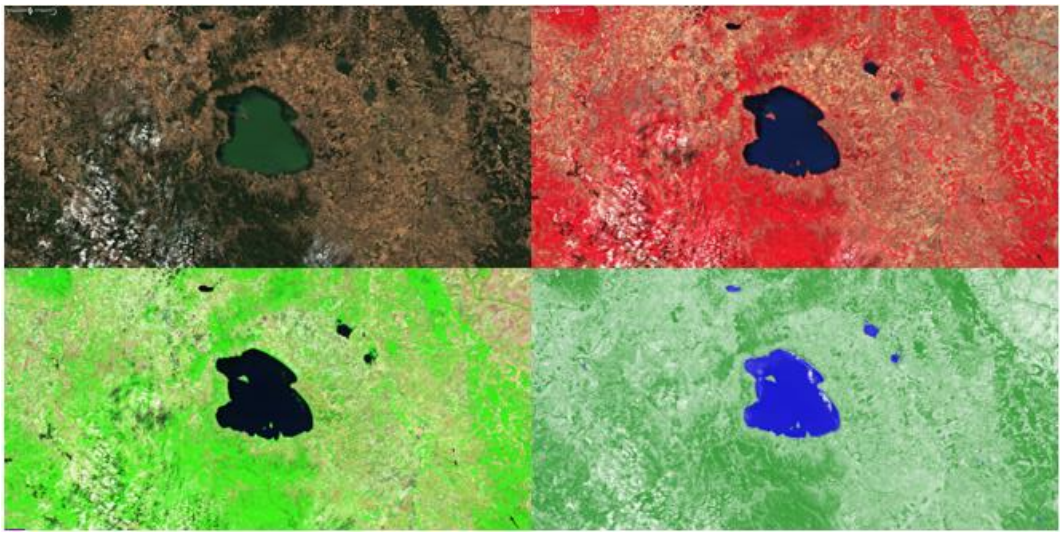


Figura 3-2. Mosaico de imágenes en diferentes espectros obtenidas mediante EO Browser.
Realizado por: Núñez, Christian, 2021.

2.2 Preprocesamiento de imágenes

Una vez descargadas las imágenes, se deben realizar una serie de pasos de preprocesamiento que son necesarios para utilizar correctamente los datos de Sentinel 2. Este proceso debe realizarse en todas las imágenes que se utilizarán.

2.2.1 Recorte y orientación

Hay que tener muy en cuenta la resolución y orientación de las imágenes que vayamos a utilizar como nuestros datos para el entrenamiento de la red neuronal, puesto que la misma solo acepta una única resolución y orientación para absolutamente todas las imágenes. La resolución de las imágenes influye en el tiempo que tardará en entrenarse la red, mientras de menor tamaño sean las imágenes, menos se tardará el proceso de entrenamiento. Hay que tener en cuenta que mientras menor sea la resolución, menor será el detalle de la imagen, lo que significa que la red neuronal perderá un porcentaje considerable de precisión.

La resolución es un parámetro más del algoritmo de entrenamiento de la red, por lo que se puede modificar a cualquier tamaño. En este caso se optó por elegir una resolución de 720x960 pixeles, un tamaño más que adecuado. Todas las imágenes deberán ser de un formato compatible con el software a usar.

2.2.2 Corrección de contraste

La mejora del contraste de una imagen en color generalmente se realiza transformando la imagen en un espacio de color en el que el brillo de la imagen es un componente, como el espacio de color L*a*b*. El ajuste de contraste se realiza solo en la capa de brillo "L *", donde la imagen se

convierte al espacio de color RGB. Las operaciones de brillo afectan la intensidad de los píxeles al tiempo que conservan los colores originales. (Matlab, 2021). En la Figura 4-2 se aprecian los diferentes tipos de corrección de contraste con los que cuenta el software Matlab.

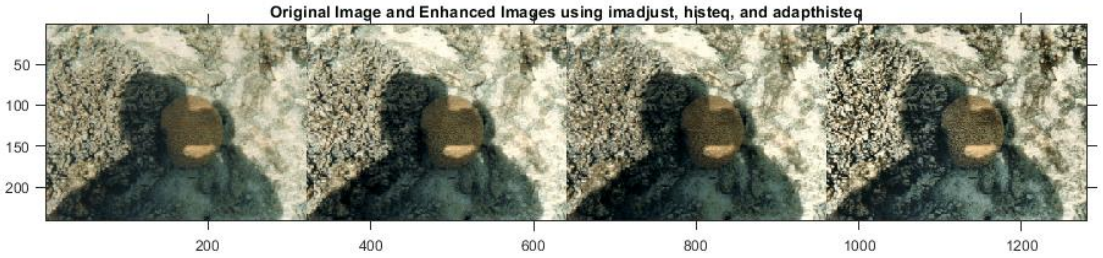


Figura 4-2. Mejora de contraste de una imagen con tres tipos de técnicas diferentes.

Realizado por: Matlab, 2021.

*2.2.2.1 Espacio de color CIE L*A*B**

Un espacio de color se puede describir como una forma de utilizar algún tipo de símbolo (como un número) para representar el color de un objeto. La Comisión Internacional de Iluminación (CIE), una organización sin fines de lucro considerada una autoridad en la ciencia de la luz y el color, define espacios de color como CIE XYZ, CIE L*C*h y CIE L*a*b*. Para comunicar y representar los colores de manera objetiva.

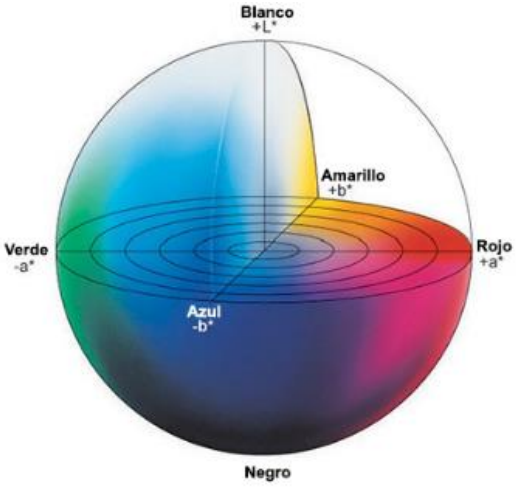


Figura 5-2. Espacio de color CIE L*A*B*.

Realizado por: Konica Minolta Sensing Americas, 2020.

El espacio de color L*a*b*, también llamado CIELAB, es uno de los espacios de color más comunes y uniformes que se utilizan actualmente para evaluar los colores de los objetos. Este espacio de color se usa ampliamente porque correlaciona constantemente los valores de color digital con la visión humana. Los investigadores y los fabricantes lo utilizan para evaluar los

atributos del color, identificar inconsistencias y representar con precisión los resultados a otros en forma digital.

El color corresponde a la percepción e interpretación subjetiva. Cuando dos personas ven el mismo objeto, pueden usar diferentes puntos de referencia y diferentes palabras para representar el mismo color, lo que puede generar confusión y desinformación dentro o en toda la cadena de suministro. Para evitar esta situación y garantizar que las muestras cumplen los criterios, los colores deben expresarse en valores numéricos objetivos.

Al clasificar los colores, se pueden expresar en términos de tono (color), brillo (luminosidad) y saturación (brillo). Establecer las escalas para estos atributos permite una representación precisa de los colores.

El espacio de color L * a * b * se basa en la teoría del color de los opuestos. Esto significa que dos colores no pueden ser rojo y verde, o amarillo y azul al mismo tiempo. Como se muestra en la Figura 5-2, L* representa el brillo, mientras que a* y b* son coordenadas de cromaticidad.

- L* = brillo
- a* = coordenadas rojo / verde (+a significa rojo, -a significa verde).
- b* = coordenadas amarillo / azul (+b significa amarillo, -b significa azul).

Los instrumentos de medición de color, como espectrofotómetros y colorímetros, pueden cuantificar fácilmente estos atributos de color. Determinan el color de un objeto en el espacio de color y muestran el valor de cada coordenada L*, a* y b*. (Konica Minolta Sensing Americas, 2020)

Figura 6-2. Ejemplo de corrección de contraste, original (izquierda), modificado (derecha).

Realizado por: Núñez, Christian, 2021.

En la Figura 6-2 podemos apreciar la corrección de contraste aplicada a una imagen satelital de un recurso hídrico, este proceso será aplicado a todas las imágenes que vamos a utilizar como datos de entrenamiento para nuestra red.

2.3 Entrenamiento de la red neural

En esta sección se muestra el procedimiento para entrenar la CNN que hará posible la segmentación semántica, para lo cual habrá que etiquetar todos los datos que hayamos obtenidos de recursos hídricos. Se crearán *datastore* de las imágenes y de las etiquetas de los datos de entrenamiento. Posteriormente se seleccionará un red neuronal y mediante el software Matlab, se creará el algoritmo de entrenamiento de la red, donde se deberá especificar los parámetros de entrenamiento.

2.3.1 Etiquetado de imágenes

La aplicación *Image Labeler* proporciona una manera fácil de crear de forma interactiva una variedad de formas para marcar como etiquetas de región de interés (ROI). Puede crear etiquetas de ROI rectangulares, polilíneas, de píxeles y poligonales y etiquetas de escena en una imagen o secuencia de imágenes.

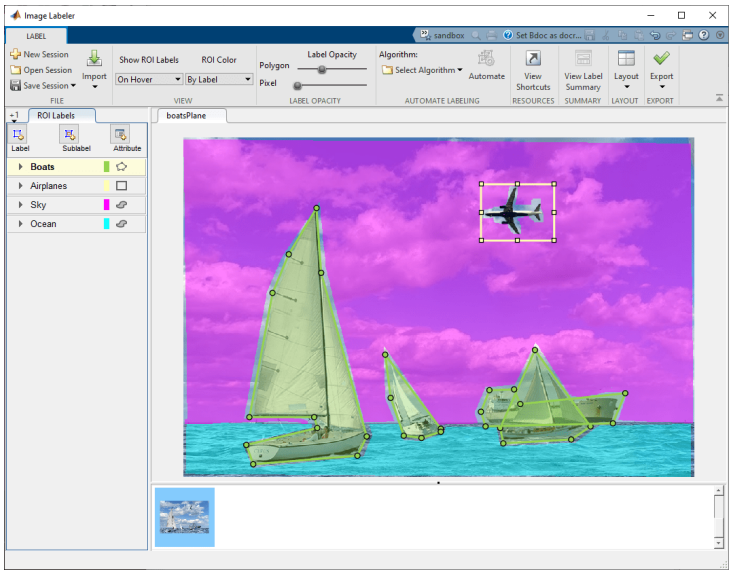


Figura 7-2. *Image Labeler* - Área de trabajo.

Realizado por: Matlab, 2021.

Se puede utilizar datos etiquetados para validar o entrenar algoritmos como clasificadores de imágenes, detectores de objetos y redes de segmentación semántica y de instancias. Hay que tener en cuenta su aplicación al elegir una herramienta de dibujo de etiquetado para crear etiquetas de retorno de la inversión.

Definiciones de etiquetas de escena y ROI:

- Una etiqueta de ROI corresponde a una región de interés rectangular, polilínea, píxel o poligonal. Estas etiquetas contienen dos componentes: el nombre de la etiqueta, como "automóviles", y la región que crea.
- Una etiqueta de escena describe la naturaleza de una escena, como "soleado". Puede asociar esta etiqueta con un marco.

La aplicación *Image Labeler*, cuya interfaz se puede apreciar en la Figura 7-2, admite todos los formatos de archivo de imagen soportados por la función *imread* y, además, admite el formato *Digital Imaging and Communication in Medicine* (DICOM). Para leer formatos de archivo adicionales, puede crear un *imageDatastore* y usar la propiedad *ReadFcn*. (Matlab, 2021)

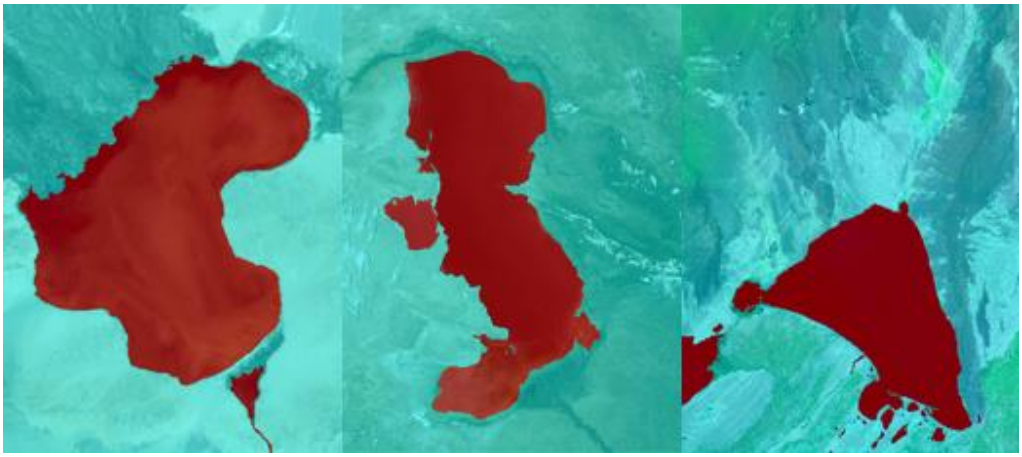


Figura 8-2. Ejemplos de lagos etiquetados.

Realizado por: Núñez, Christian, 2021.

En la Figura 8-2 se muestra algunas imágenes ya preprocesadas y etiquetadas, como se puede observar, hay únicamente dos etiquetas, la de color cian corresponde a la etiqueta de tierra firme y la de color rojo corresponde a la etiqueta de agua.

2.3.2 Datastore

La función *datastore* crea un almacén de datos, que es un repositorio de colecciones de datos que son demasiado grandes para caber en la memoria. Un almacén de datos le permite leer y procesar datos almacenados en varios archivos en un disco, una ubicación remota o una base de datos como una sola entidad. Si los datos son demasiado grandes para caber en la memoria, puede administrar la importación incremental de datos, crear una matriz alta para trabajar con los datos o usar el almacén de datos como entrada a *mapreduce* para su procesamiento posterior.

La función *ImageDatastore* nos ayuda a administrar una colección de archivos de imagen, donde cada imagen individual cabe en la memoria, pero la colección completa de imágenes no encaja necesariamente. Se puede crear un almacén de datos de imágenes mediante la función *imageDatastore*, especificar sus propiedades y luego importar y procesar los datos mediante funciones de objeto. *PixelLabelDatastore* en cambio, nos sirve para crear bancos de datos de etiquetas de píxeles para la segmentación semántica. (Matlab, 2021)

2.3.3 Resnet-18

ResNet-18 es una red neuronal convolucional con una profundidad de 18 capas. Se puede cargar con una versión previamente entrenada de la red que ha sido entrenada con más de un millón de imágenes de la base de datos ImageNet. La red previamente entrenada puede clasificar una imagen en 1000 categorías de objetos. (Matlab, 2021)

Tabla 1-2: Arquitectura de ResNet-18.

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	$7 \times 7, 64, stride\ 2$
conv2_x	$56\times 56 \times 64$	$3 \times 3 \max pool, stride\ 2$ $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	$7 \times 7\ average\ pool$
fully connected	1000	$512 \times 1000\ fully\ connections$
softmax	1000	

Fuente: Napoletano, Piccoli, Schettini, 2018.
Realizado por: Núñez, Christian, 2021.

En la Tabla 1-2 se muestra la arquitectura de una red Resnet-18. La función principal de una red residual, ResNet, es aumentar el número de capas mediante la introducción de conexiones residuales. Esta capa se pasa directamente a la siguiente capa a través de accesos directos o *shortcuts*, lo que mejora el proceso de aprendizaje. De esta forma, el modelo profundo alcanza un nivel de error más bajo que el modelo superficial. En este caso, los atajos se encuentran entre capas convolucionales sucesivas. El bloque de capa que contiene las conexiones de acceso directo se denomina bloque de capa de descanso o capa ResNet. La Figura 9-2 presenta, en diagrama de bloques, la estructura de una Resnet-18 empleada para clasificación de imágenes.

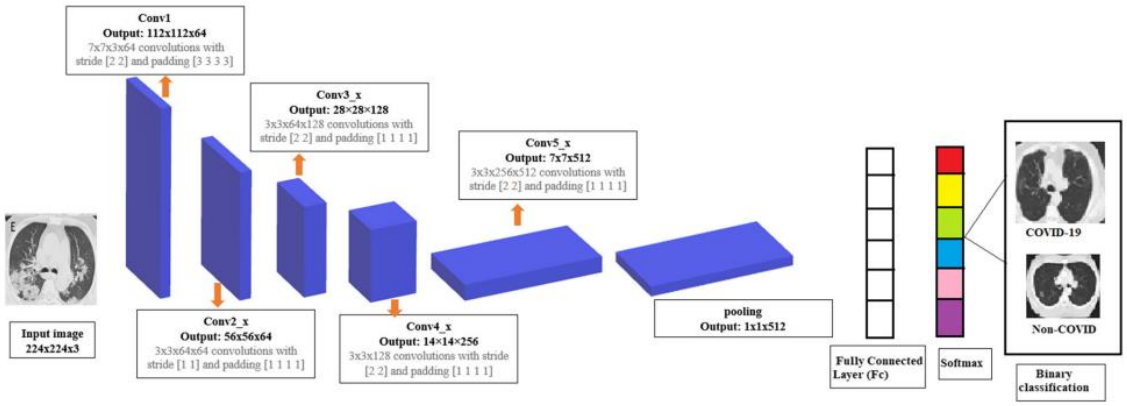


Figura 9-2. Diagrama de bloques que describe la arquitectura de la ResNet-18.

Realizado por: Ahuja, S.; et al. 2021.

Una capa ResNet consta de una capa de convolución 2D, una capa *ReLu*, una capa de normalización y una capa aditiva. El funcionamiento de esta red implica transformar la señal original x con $F(x)$, pasarla por la capa convolución-ReLu-convolución y finalmente agregarle la señal original. De esta forma, el bloque ResNet realiza la transformación $H(x) = F(x) + x$.

La idea principal de una ResNet es que no es muy difícil optimizar la $F(x)$ residual y optimizar la señal libre de referencia original. Las conexiones de acceso directo se pueden realizar mediante redes neuronales de retroalimentación con conexiones de acceso directo. Estas son conexiones que omiten una o más capas. En el caso que se muestra en la Figura 10-2, la conexión de acceso directo está dedicada al mapeo de identidad y su salida se agrega a la salida de la capa de pila. Estos saltos no introducen ninguna complejidad computacional ni generan nuevos parámetros. (González González, 2020)

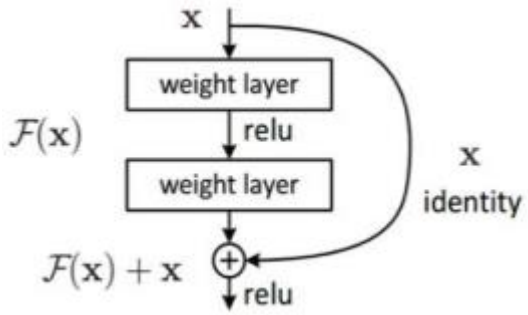


Figura 10-2. Contorno principal del bloque de características residuales.

Realizado por: González González, M. 2020.

Los modelos ResNet fueron propuestos en "*Deep Residual Learning for Image Recognition*". Aquí tenemos las 5 versiones de modelos ResNet, que contienen 18, 34, 50, 101, 152 capas respectivamente. (Pytorch Team, 2021). Las arquitecturas de modelos detalladas se pueden encontrar en la Tabla 2-2.

Tabla 2-2: Arquitectura de diferentes ResNet.

Model structure	Top-1 error	Top-5 error
resnet18	30,24	10,92
resnet34	26,70	8,58
resnet50	23,85	7,13
resnet101	22,63	6,44
resnet152	21,69	5,94

Fuente: Pytorch Team, 2021.

Realizado por: Nuñez, Christian, 2021.

2.3.4 Entrenamiento

Ahora que hemos cubierto los hechos de la clasificación de redes neuronales, la pregunta más importante es: ¿Cómo las entrenamos? ¿Cómo aprenden? Si bien parece difícil considerar estas preguntas a priori, en la práctica la respuesta es bastante simple y se deriva de las siguientes razones: Como se muestra en la Figura 5-1, la red consta de entradas x_i (dada al sistema porque es un valor fijo), los pesos w_i y el sesgo b_i, son estos valores los que determinan la importancia de las entradas y la forma en que los resultados de las operaciones computacionales activan las neuronas. En otras palabras, estos dos últimos parámetros se pueden cambiar y ajustar, y son los parámetros que se cambiarán a lo largo del proceso de entrenamiento hasta que se alcance un conjunto de valores que maximice la precisión del clasificador. Por lo tanto, necesitamos definir un proceso matemático que refleje este objetivo. Este proceso se denomina *backpropagation* y actualiza los pesos de la red para que se dirijan a los mejores valores para optimizar la tasa de aciertos de clasificación. (Hormigo Ruiz, 2017). Hay muchas formas de entrenar redes neuronales y otros tipos de clasificadores, aquí se presentan los más utilizados:

- Resustitución: Esto implica el uso de todas las muestras en el conjunto de datos para entrenamiento, prueba y validación. Este es un enfoque optimista y proporciona una tasa de aciertos del 100% en los datos existentes, pero no proporciona una tasa de generalización para el modelo.
- Hould-Out: Implica dividir el conjunto de datos en dos o tres subconjuntos (entrenamiento, validación y prueba). Estos representan el 70%, 15% y 15% del tamaño del conjunto de datos, respectivamente. En general, las estimaciones obtenidas serán inferiores a la precisión de clasificación real.
- Validación cruzada: Implica dividir el conjunto de datos en k subconjuntos, de modo que los conjuntos de datos $k - 1$ se utilicen para el entrenamiento y el resto para las pruebas y se

itera hasta obtener los k resultados de la clasificación. El ratio final se calcula como la media de los resultados obtenidos.

- Bootstraping: Se realiza un submuestreo alternativo del conjunto de datos para obtener n conjuntos de datos de entrenamiento y un conjunto de datos de prueba, y se obtiene la tasa de acierto de cada uno de los n modelos generados usando el mismo conjunto de prueba para todos ellos, siendo la tasa de acierto la tasa promedio de las n tasas de acierto.

2.3.5 Data augmentation

El *Data augmentation* es un método para aumentar la diversidad de un conjunto de datos de entrenamiento mediante la aplicación de transformaciones aleatorias (pero realistas), como la rotación de imágenes. Existen varios métodos, la creación de diferentes datos a partir de los mismos datos puede resultar muy útil. Esto también es para evitar que ocurra un *overfitting* o sobreajuste de la red. No es necesario utilizar todas estas técnicas. Esto depende de si es necesario o no atendiendo al resultado que se desea lograr. (Roman y Ruiz, 2021)

2.3.5.1 Rotación

Como su nombre indica, su función principal es la capacidad de voltear una imagen en una y dos dimensiones. Por lo general, la imagen se gira solo horizontalmente. Hay otros tipos que le permiten rotar la imagen 90, 180 o 270 grados.

2.3.5.2 Corte

Se incluyen alto y ancho, lo que implica ajustar el tamaño de la imagen. La imagen se reducirá en los ejes X e Y, y estos dos nuevos valores pertenecerán a estos valores originales de alto y ancho. Otra advertencia sobre esta técnica es que nos permite encoger una imagen sin cambiar el número de píxeles de la imagen original.

2.3.5.3 Saturación de color y transformación gamma

La primera técnica nos permite cambiar ciertos parámetros de color y ver cuánto cambian en respuesta a la exposición a ciertos cambios en la intensidad de la luz. La transformación gamma se basa en ajustar el contraste de una imagen.

2.3.6 Hiperparámetros

Los hiperparámetros de la red son los valores de configuración que se utilizan en el proceso de entrenamiento. Son valores que generalmente no se toman de los datos, por lo que generalmente los proporciona el científico de datos. No es posible conocer de antemano los valores óptimos de los hiperparámetros para un problema en particular. Por lo tanto, las personas necesitan usar valores generales, reglas generales, valores que han sido válidos antes para problemas similares

o encontrar la mejor opción a través de prueba y error. Es recomendable encontrar hiperparámetros mediante validación cruzada.

Al entrenar un modelo de aprendizaje automático, el valor del hiperparámetro se establecerá para obtener el parámetro. (Rodríguez 2019). A continuación, en la Tabla 3-2 se muestran algunos ejemplos de hiperparámetros utilizados para entrenar modelos.

Tabla 3-2: Descripción de hiperparámetros.

Campo	Descripción
Epoch	Número de época. Las épocas corresponden a canales de datos completos.
Iteration	Número de iteraciones. Una iteración corresponde a un lote pequeño.
Time Elapsed	Tiempo transcurrido en horas, minutos y segundos.
Mini-batch Accuracy	Precisión de clasificación para pequeños lotes de datos.
Validation Accuracy	Verifica la precisión de clasificación de los datos. Si no especifica ningún dato de validación, la función no mostrará este campo.
Mini-batch Loss	Pérdida de lotes pequeños. Si la capa de salida es un objeto *ClassificationOutputLayer*, la pérdida es la pérdida de entropía cruzada del problema de clasificación multiclase para clases mutuamente excluyentes.
Validation Loss	Pérdida de datos de validación. Si la capa de salida es un objeto *ClassificationOutputLayer*, la pérdida es la pérdida de entropía cruzada para problemas de clasificación multiclase con clases mutuamente excluyentes. Si no se especifican datos de validación, la función no mostrará este campo.
Base Learning Rate	Tasa de aprendizaje básica. El software multiplica el factor de tasa de aprendizaje para cada capa por este valor.

Fuente: Matlab, 2021.

Realizado por: Núñez, Christian, 2021.

2.4 Segmentación

En este apartado implementaremos la red neuronal entrenada en un algoritmo de Matlab que nos permitirá introducir imágenes desde el computador para su posterior segmentación. Una vez realizada la segmentación se procederá a extraer una máscara binaria a partir de la etiqueta "agua" de la imagen etiquetada.

2.4.1 Reconocimiento de escena

Como comprobación rápida, ejecutamos la red entrenada en una imagen de prueba.

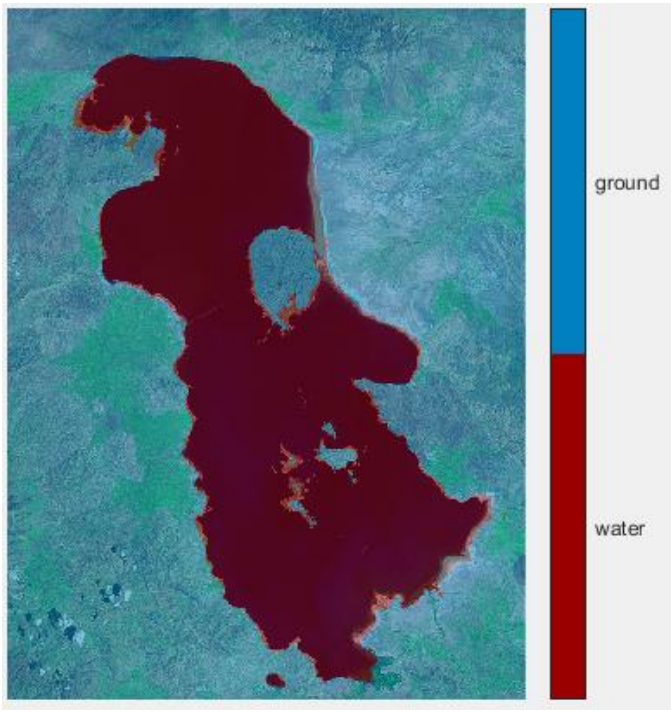


Figura 11-2. Testeo de la red entrenada sobre una imagen satelital.

Realizado por: Núñez, Christian, 2021.

Comparamos los resultados obtenidos en la Figura 11-2 con el *ground truth* almacenada en *pxdsTest*. Las regiones verde y magenta resaltan las áreas en las que los resultados de la segmentación difieren de la verdad terrestre esperada, véase Figura 12-2.

Visualmente, los resultados de la segmentación semántica se superponen bien, sin embargo, los detalles más pequeños como las islas, no son tan precisos. La cantidad de superposición por clase se puede medir utilizando la métrica de intersección sobre unión (IoU), también conocida como índice de Jaccard.

Tabla 4-2: Métrica de intersección sobre unión.

Classes	IoU
'water'	0.93665
'ground'	0.96957

Realizado por: Núñez, Christian, 2021.

Como se muestra en la Tabla 4-2, a pesar de existir ciertas fallas en el etiquetado de la imagen, ambas clases presentan una alta puntuación de IoU, lo que se entiende en una alta precisión de clasificación.

Figura 12-2. Comparación de la segmentación hecha por la red contra el *ground truth*.

Realizado por: Núñez, Christian, 2021.

2.4.2 Obtención de máscara

En una imagen binaria, cada píxel tiene uno de sólo dos valores discretos: 1 o 0. En la mayoría de los casos se interpretan los píxeles con el valor 1 como pertenecientes a una región de interés y los píxeles con el valor 0 como fondo. Las imágenes binarias se utilizan con frecuencia junto con otros tipos de imágenes para indicar qué partes de la imagen se deben procesar. (Matlab, 2021). La Figura 13-2 muestra una imagen binaria con una vista de cerca de algunos de los valores de píxeles.

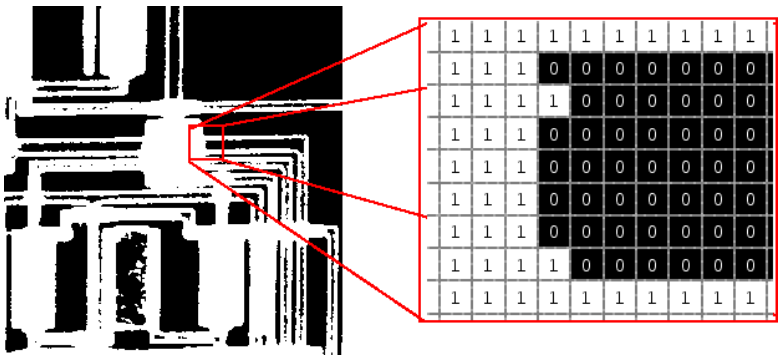


Figura 13-2. Representación de una imagen binaria a nivel de píxeles.

Realizado por: Matlab, 2021.

En la Figura 14-2 se muestra una imagen binarizada obtenida a partir del etiquetado realizado por la red entrenada, para este caso, el área de interés es la clase agua. Esta imagen binaria se utilizará como una máscara en el apartado de post procesamiento, donde se emplearán técnicas PDI.

Figura 14-2. Mascara obtenida por la binarización de la clase '*water'*.
Realizado por: Núñez, Christian, 2021.

2.5 Post Procesamiento

Tras haber obtenido la máscara de la imagen original, se procederá a aplicar diversas técnicas de PDI para obtener mejores resultados en la selección de área de interés. Las técnicas empleadas se basan en operaciones morfológicas y contornos activos.

2.5.1 Operaciones morfológicas

La morfología matemática es una herramienta muy utilizada en el procesamiento de imágenes. Las operaciones morfológicas pueden simplificar los datos de las imágenes, preservar las características básicas y eliminar los aspectos irrelevantes. El papel de la morfología matemática es evidente en el reconocimiento y descomposición de objetos, extracción de características, ubicación de defectos e incluso defectos de la línea de montaje, que están relacionados en gran medida con la forma. (Aranda Delgadillo et al., 2017). La morfología matemática se puede utilizar para, entre otros, los siguientes propósitos:

- Preprocesamiento de imágenes (eliminación de ruido, simplificación de formas).
- Resaltar la estructura de los objetos (extracción del esqueleto, marcado de objetos, envolvente convexa, acercar, alejar).

- Descripción cualitativa de objetos (área, perímetro, diámetro, etc.).
- Visión artificial y manipulación morfológica de imágenes binarias.

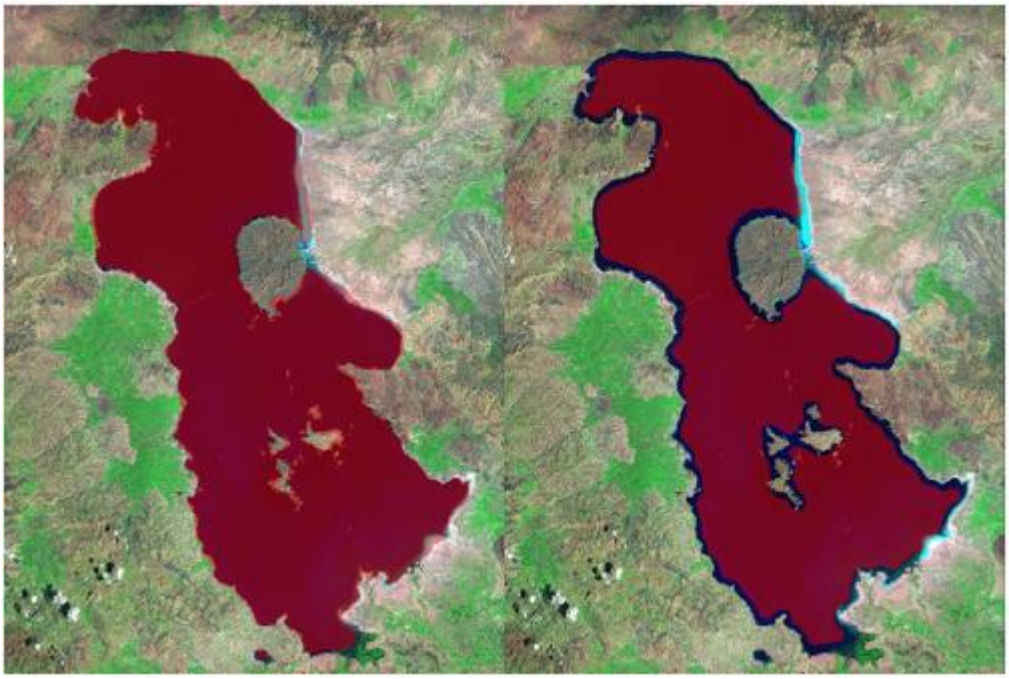


Figura 15-2. Mascara original (izquierda) y su apertura y erosionado (derecha).
Realizado por: Núñez, Christian, 2021.

En la Figura 15-2 podemos ver la máscara superpuesta en la imagen original y el resultado que se obtiene luego de aplicarse operaciones morfológicas, primero se le aplico un proceso de apertura y consecutivamente un proceso de erosión, dando como resultado una máscara que excluye casi en su totalidad las orillas del recurso. Esta nueva máscara no servirá para aplicar el último paso del post procesado, un proceso denominado: contornos activos o *active contours*.

2.5.2 Contornos activos

Las contornos activos (generalmente llamadas *Snakes*) consisten en curvas elásticas aproximadas que se colocan en la imagen y se deforman desde su forma inicial para representar la región de interés en la escena. Se basan en modelos físicos de elastómeros. Por tanto, su evolución en el tiempo y el espacio viene dada por parámetros de elasticidad y rigidez. En el caso de la detección de bordes, además de estos parámetros, intervienen fuerzas en la deformación del contorno inicial. Estas fuerzas pueden estar dadas por la información que proporciona la imagen (fuerzas internas) o por elementos externos de la propia imagen (fuerzas externas), como dejar que el contorno final tenga un área específica. (Bastida Jumilla et al., 2008)

Según algunos autores, el modelo utilizado para la detección de contornos debería tener las siguientes propiedades:

- Determinar contornos globales que reproduzcan mejor los contornos obvios en la imagen.
- Capacidad de extraer formas suaves (contornos regulares).
- Respetar algunas singularidades: ángulos, esquinas, etc.
- Robustez del algoritmo: parámetros mínimos establecidos para la estabilidad y la convergencia.
- Posible adaptabilidad para encontrar varios contornos simultáneamente.

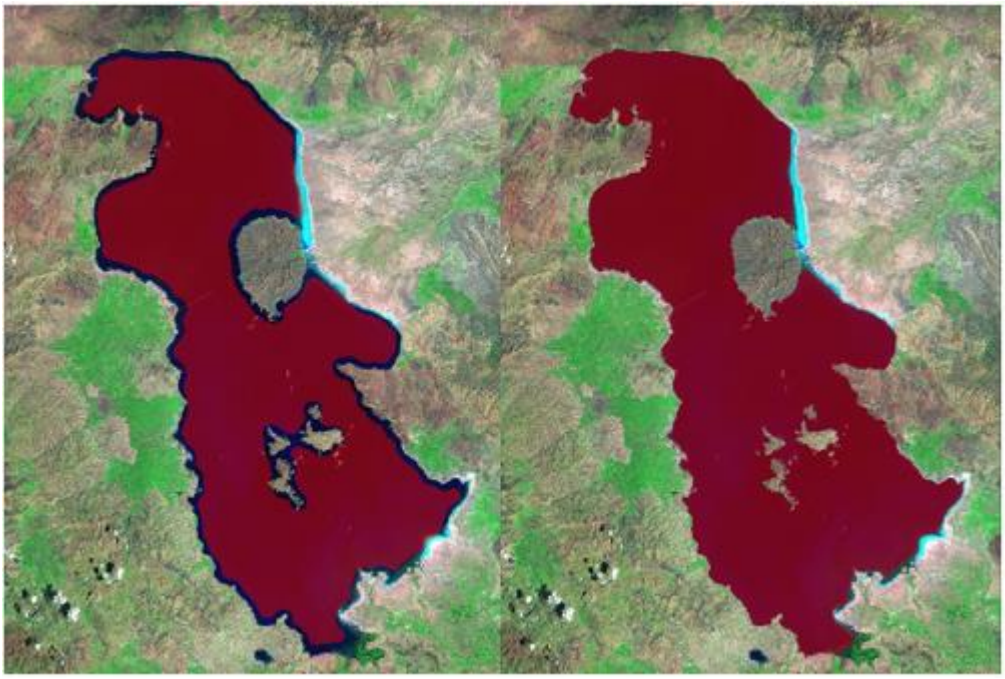


Figura 16-2. Resultado de aplicar contornos activos a la máscara obtenida previamente.
Realizado por: Núñez, Christian, 2021.

La Figura 16-2 muestra los resultados de aplicar contornos activos en la máscara obtenida tras aplicarle operaciones morfológicas, el resultado muestra una precisión bastante superior a la segmentación realizada por la red neuronal entrenada por si sola.

2.6 Obtención de área

El paso final consiste en determinar el área del recurso hídrico en km^2. Para obtener el área únicamente habrá que tomar la máscara obtenida luego del post procesado, que tendrá un formato binario, y realizar el conteo de los píxeles activos, es decir, que tengan el valor de 1. Un píxel activo representará una pequeña área de la imagen en metros cuadrados, para determinar esta área habrá que tener en cuenta la resolución espacial de dicha imagen, la que nos indicará la distancia de un lado del píxel que, al ser un cuadrado, su área será el resultado de calcular la resolución

espacial al cuadrado. Una vez obtenida el área de cada píxel individual, para obtener el área final del recurso únicamente habrá que multiplicar el área del píxel por el número de píxeles activos.

En la Figura 17-2 se muestra el resultado final obtenido del algoritmo, se puede apreciar al recurso hídrico segmentado, contorneado y con el área obtenida mostrada sobre el mismo.

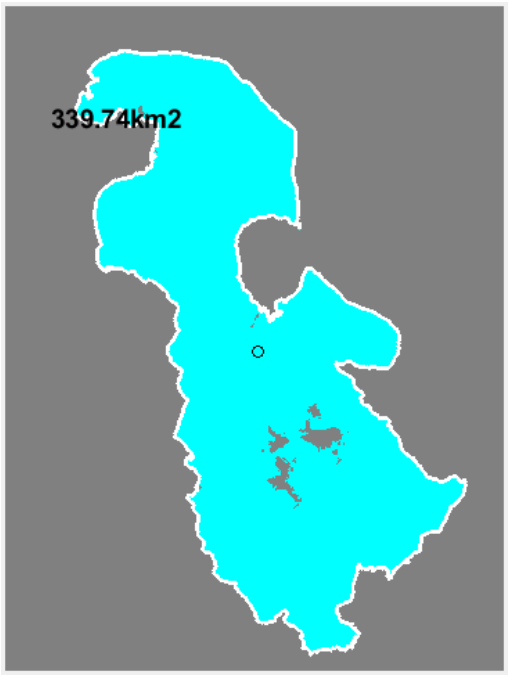


Figura 17-2. Cálculo del área del recurso hídrico.

Realizado por: Núñez, Christian, 2021.

CAPÍTULO III

3. RESULTADOS

En este capítulo se muestran los resultados obtenidos de la segmentación realizada por la red neuronal entrenada y los resultados obtenidos por los algoritmos de postprocesamiento, haciendo evidente la mejora en la precisión de la segmentación. También se hace una comparación del software diseñado con otras alternativas de cálculo de área de interés.

3.1 Recopilación y selección de imágenes satelitales

En esta etapa de recopilación de imágenes de prueba, se realizó una búsqueda de diferentes conjuntos de datos que podrían usarse para nuestros propósitos. Para fines más pedagógicos y para poder realizar nuestro trabajo según una aplicación específica, seleccionamos estratégicamente una serie de fotografías satelitales. Estas imágenes se organizarán e integrarán para formar un conjunto de datos de entrada como punto de partida.

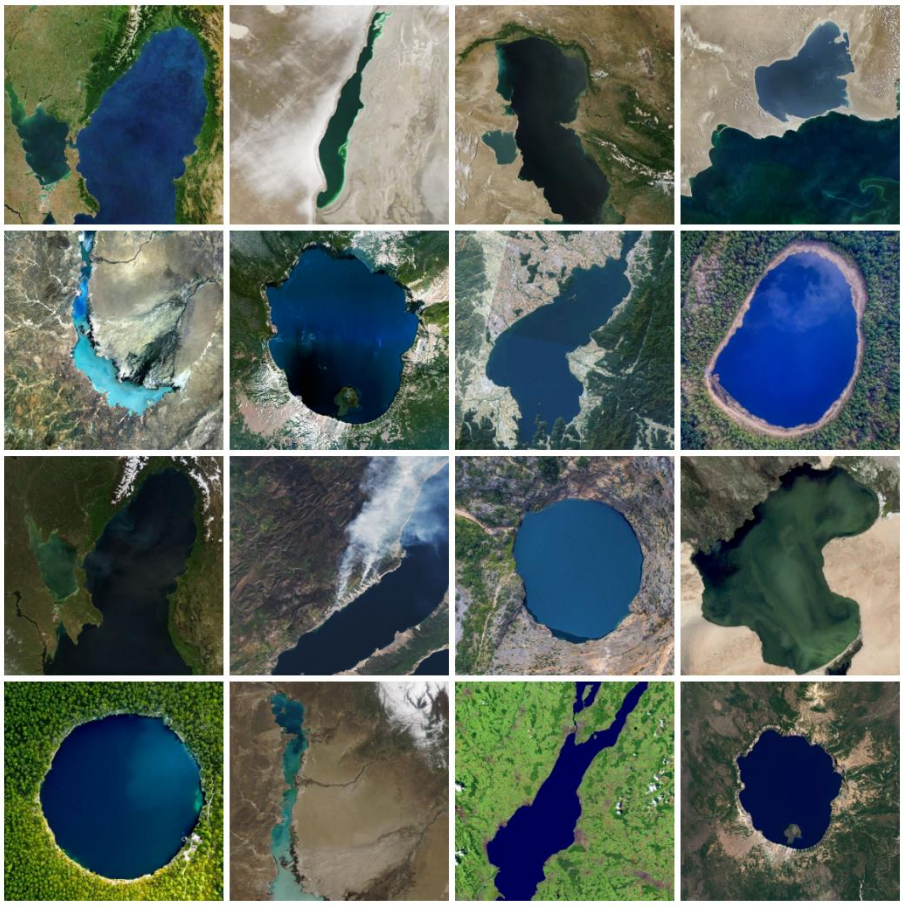


Figura 1-3. Imágenes de muestra.

Realizado por: Núñez, Christian, 2021.

En la Figura 1-3 se muestra un conjunto de imágenes que forman parte del total de imágenes de recursos hídricos utilizado como conjunto de datos para el entrenamiento de la red neuronal, todas las imágenes fueron obtenidas del sitio Sentinel Hub, por medio de la herramienta EO Browser.

3.2 Preprocesamiento de imágenes

Esta fase cubre todos los métodos aplicados al conjunto de datos de entrada para mejorar ciertas características y eliminar fallas del conjunto de imágenes. Los métodos aplicados se basan en los vistos anteriormente en la literatura como: filtrar, enfocar, iluminar, recortar, redimensionar y reasignar niveles de intensidad para mejorar la imagen. Como resultado de esta etapa, obtenemos una serie de imágenes procesadas con mejores características. Esto facilita la realización de etapas posteriores. En la Figura 2-3 se muestran las imágenes originales junto a las imágenes corregidas, siendo evidente el cambio de contrastes y saturación de colores.

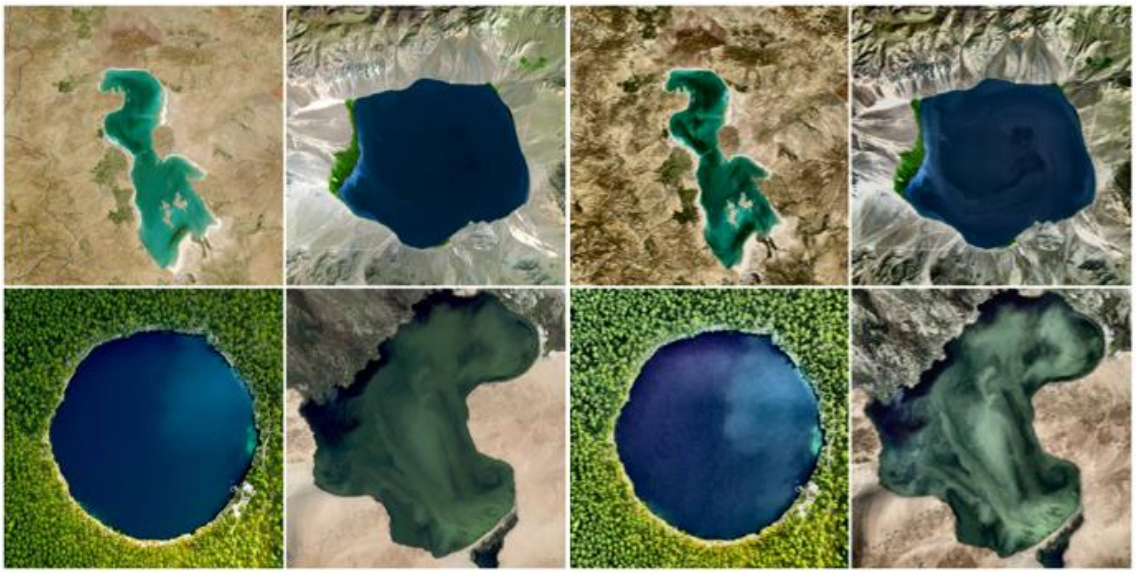


Figura 2-3. Ejemplo de imágenes preprocesadas.
Realizado por: Núñez, Christian, 2021.

3.3 Etiquetado de imágenes

Después de procesar y mejorar las características de la imagen, puede comenzar la fase de etiquetado. Dado que ejecutar un modelo de aprendizaje profundo requiere una gran cantidad de muestras, una de las posibles opciones es descargar datos etiquetados de Internet. Como se mencionó anteriormente, el objetivo de este proyecto es ejecutar el modelo en una aplicación específica, por lo que realizaremos esta fase manualmente utilizando el software MATLAB. Usando la aplicación *Image Labeler* de MATLAB podemos clasificar cada píxel en diferentes categorías de una manera muy dinámica y sencilla. Esto nos permite controlar qué categorías distinguir dentro de una imagen. Una vez que se haya generado un conjunto de imágenes

etiquetadas, las combinamos en un banco de datos que contenga la ubicación de cada imagen. Hay dos conjuntos claramente identificados: uno es el *datastore* de las imágenes originales y el otro es el *datastore* de las imágenes etiquetadas.

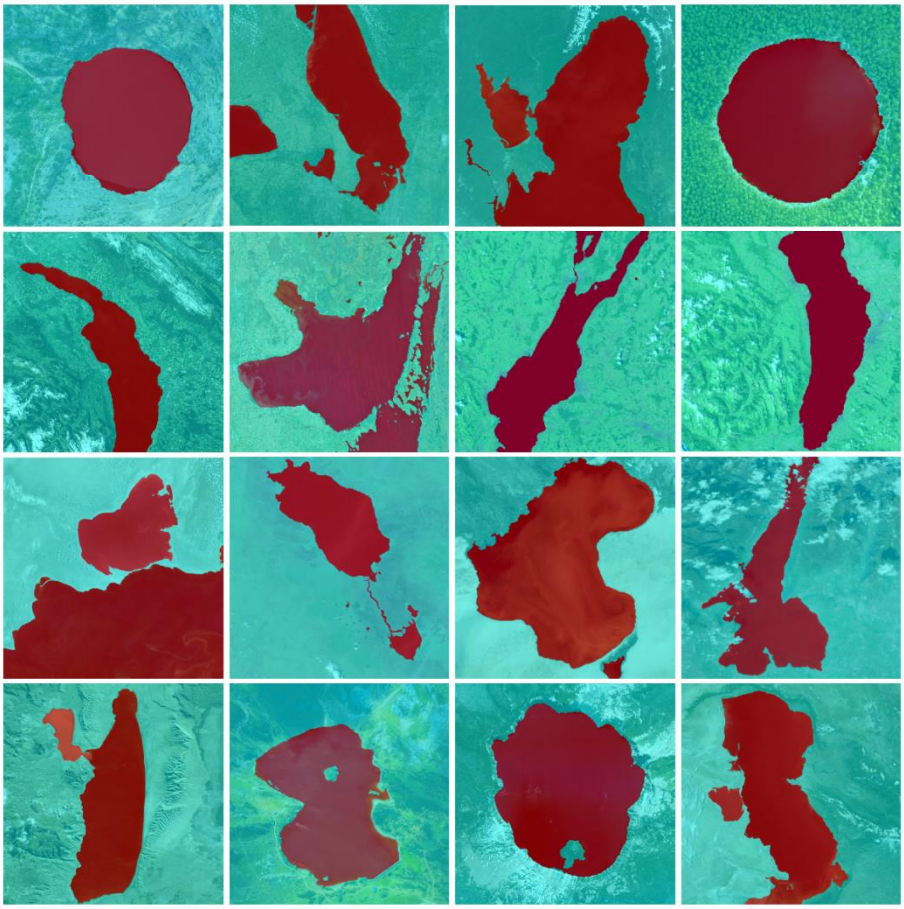


Figura 3-3. Imágenes de muestra respectivamente etiquetadas.

Realizado por: Núñez, Christian, 2021.

En la Figura 3-3 se muestra parte del *datastore* de las imágenes etiquetadas, como ya se había mencionado, hay únicamente dos etiquetas; 'water' o agua, y 'ground' o tierra firme. Los datos de las etiquetas nos servirán para entrenar la red, ya que la misma necesita aprender a diferenciar entre clases, por lo que el etiquetado debe ser lo más preciso posible.

3.4 Entrenamiento

Una vez almacenados los dos conjuntos de imágenes, las originales y las etiquetas, podemos iniciar la fase de entrenamiento del modelo de segmentación semántica previamente definido. El proceso de entrenamiento de un modelo de segmentación semántica basado en la red Resnet-18 implica calcular y ajustar los pesos de cada entrada de cada nodo o neurona de la red neuronal para minimizar la función de costo o función de pérdida que mide la salida del modelo. El entrenamiento se realiza en un conjunto arbitrario de imágenes y la precisión se evalúa al final

del entrenamiento del modelo. El entrenamiento de un modelo requiere parámetros como el número de iteraciones (épocas, el número de veces que se entrena todo el conjunto de datos), la tasa de aprendizaje, el número de iteraciones y el tamaño del lote (el número de muestras utilizadas en las iteraciones). Estos serán determinados según nuestras necesidades. La elección correcta de los parámetros producirá mejores resultados y una mayor precisión al final del entrenamiento. Otro factor a considerar al elegir parámetros es el uso de recursos informáticos, ya que se necesita mucho procesamiento y memoria cuando se procesan imágenes de alta resolución y espacios de color RGB. Para realizar el entrenamiento, utilizamos los siguientes recursos informáticos, como se muestra en la Tabla 1-3.

Tabla 1-3: Recursos informáticos.

CPU:	Intel(R) Core (TM) i7-2600 CPU @ 3.40GHz (8 CPUs), ~3.4GHz
RAM:	8192MB
GRÁFICOS:	Intel(R) HD Graphics Family

Realizado por: Núñez, Christian, 2021.

La implementación de un algoritmo de aprendizaje profundo requiere una alta potencia de procesamiento, especialmente en la fase de entrenamiento. Por lo tanto, a la hora de elegir los parámetros, se deben tener en cuenta dos cuestiones: una es utilizar los recursos informáticos existentes y la otra es hacer que el modelo alcance una precisión aceptable. Lograr un equilibrio entre estos dos factores es el principal objetivo de la selección de los parámetros de entrenamiento. En la Tabla 2-3 se muestran los parámetros utilizados para el entrenamiento de nuestra Resnet-18.

Tabla 2-3: Parámetros de entrenamiento de la red.

PARÁMETRO	VALOR
LearnRateDropPeriod	10
LearnRateDropFactor	0.3
Momentum	0.9
InitialLearnRate	1e-3
L2Regularization	0.005
MaxEpochs	30
MiniBatchSize	8
VerboseFrecuncy	2
ValidationPatience	4

Realizado por: Núñez, Christian, 2021.

En la Gráfico 1-3 se aprecia el proceso de entrenamiento el cual se ejecutó con una duración de 301 minutos y 28 segundos, obteniendo a una exactitud máxima de 98.79%. Este entrenamiento se ejecuta con una tasa de aprendizaje alta, debido al elevado número de repeticiones, lo que da como resultado una alta precisión, con una potencia computacional baja, razón por la cual el entrenamiento se prolongó por demasiado tiempo.

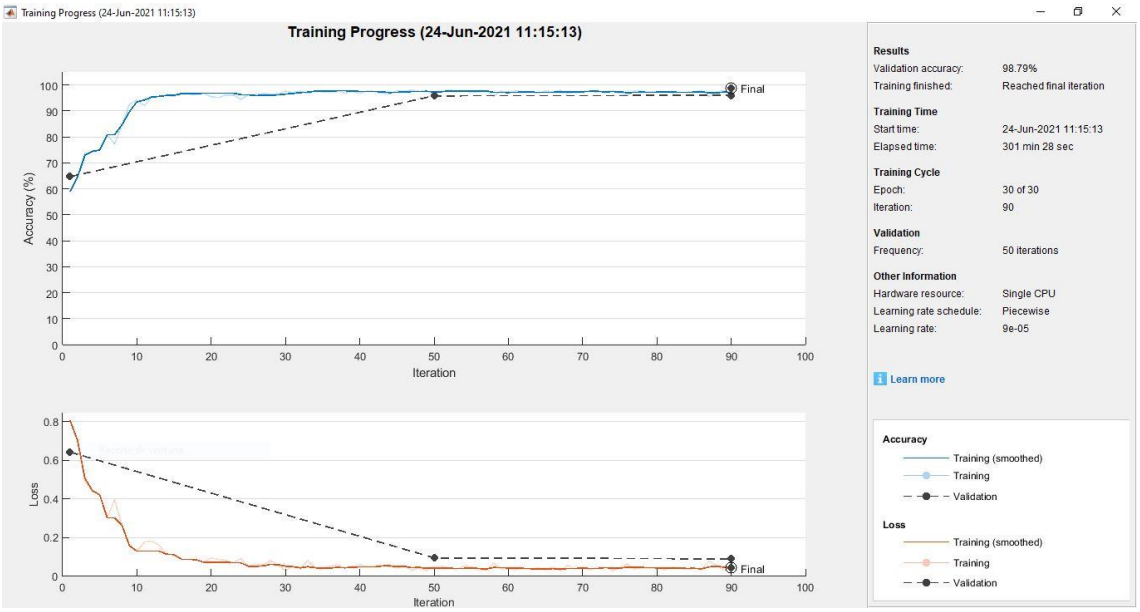


Gráfico 1-3. Precisión y perdida del entrenamiento de la red.

Realizado por: Núñez, Christian, 2021.

3.5 Evaluación de la red

Una vez que se ha seleccionado un modelo de segmentación semántica entrenado, se almacena en la memoria para que la prueba pueda continuar ejecutándose posterior al entrenamiento. Las pruebas se realizan utilizando un conjunto de datos diferente o un conjunto de imágenes satelitales "desconocidas" para nuestra red. Estas pruebas nos permiten evaluar la precisión del modelo, la confiabilidad de los resultados y detectar efectos indeseables, como la imposibilidad de analizar nuevos datos de entrada y generar clasificaciones erróneas.

Las imágenes utilizadas como test tienen su propio *Ground Truth*. De esta forma, podemos calcular la métrica IoU. En cualquier caso, de forma cualitativa, se muestran algunos resultados de segmentación obtenidos con las imágenes de prueba originales. La Figura 4-3 muestra los resultados de la segmentación individual para cinco ejemplos seleccionados del conjunto de prueba.

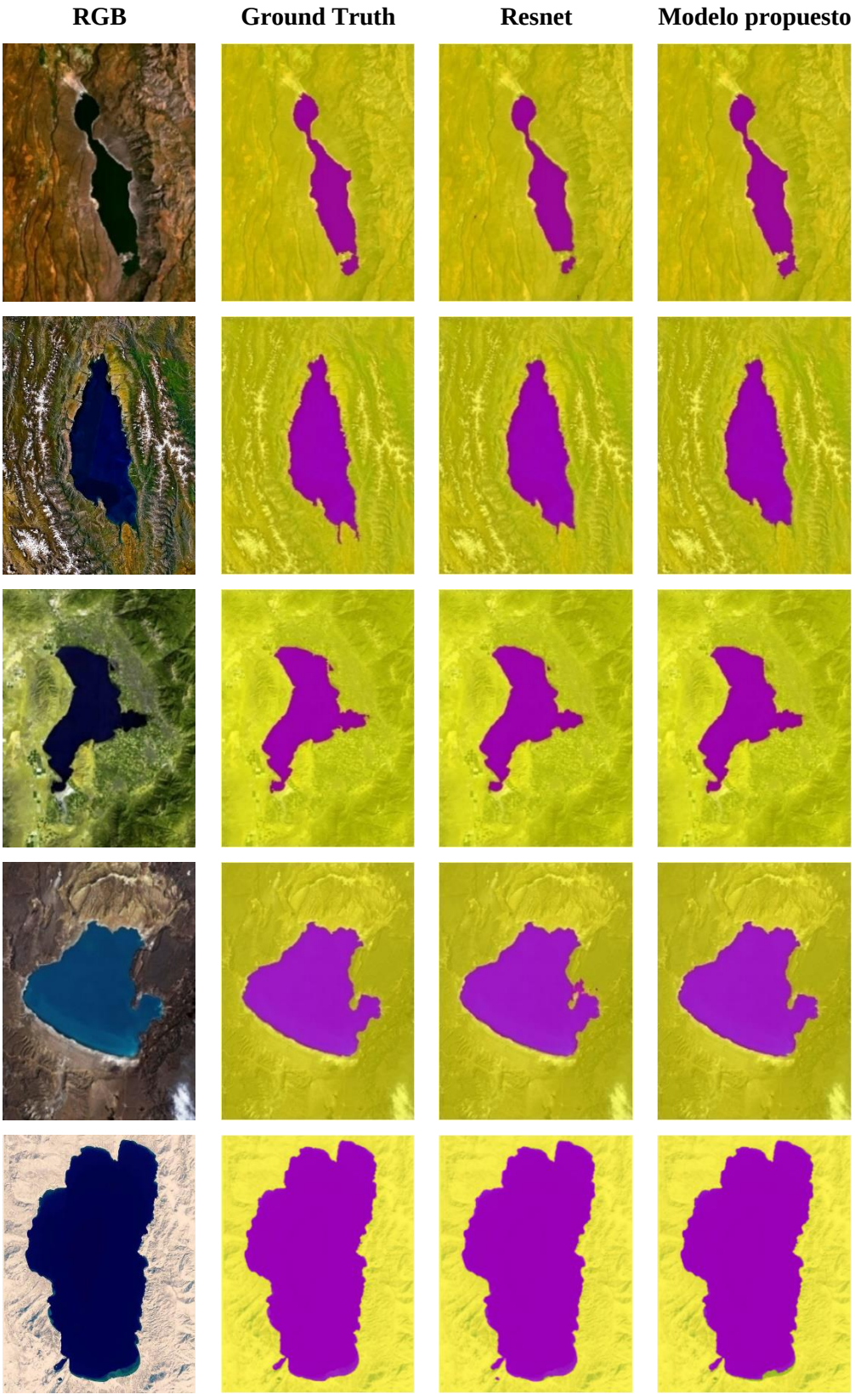


Figura 4-3. Comparativa de resultados obtenidos.

Realizado por: Núñez, Christian, 2021.

3.6 Pruebas y resultados

En esta sección presentamos los resultados obtenidos mediante un modelo de segmentación semántica basado en la arquitectura ResNet. Para ello, el modelo ha sido previamente entrenado, probado y validado. Para evaluar los objetivos a alcanzar, se presentarán una serie de resultados de imágenes.

Se utilizan cuatro métricas para evaluar el modelo.

- Precisión global: este es el porcentaje de píxeles clasificados correctamente, independientemente de la categoría y el número total de píxeles.
- Precisión promedio: representa el porcentaje de píxeles correctamente identificados en cada categoría. La precisión de cada categoría es la relación entre los píxeles clasificados correctamente y el número total de píxeles en esa categoría. Esto se conocía anteriormente como imagen etiquetada.
- Coeficiente Jaccard: conocido como IoU (*Intersection over union*), mide la proporción de píxeles clasificados correctamente con respecto al número total de píxeles reales y previstos en una categoría.
- Puntuación de coincidencia de contorno: *BFScore*, que calcula la coincidencia de contorno entre la segmentación prevista y la real.

A través de un conjunto de datos de prueba agrupados en un *datastore*, se evalúa el modelo de segmentación semántica y se genera un conjunto de objetos igual a la imagen de prueba. Este nuevo conjunto de datos contendrá objetos y cada píxel se clasificará en una de las categorías. Se genera el tipo de objeto clasificado. (Carrillo Velarde, 2021). Las imágenes previamente etiquetadas se utilizan para el entrenamiento (resultados esperados), se aplican los objetos generados por el modelo de segmentación (resultados reales) y se comparan uno a uno para generar los indicadores detallados en la Tabla 3-3.

Tabla 3-3: Métricas de evaluación del modelo de segmentación semántica.

Precisión Global	Precisión Promedio	Coeficiente Jaccard	BFScore
0.97138	0.97019	0.92213	0.81844

Realizado por: Núñez, Christian, 2021.

Como se muestra en la Tabla 3-3, cuando ejecutamos el modelo de segmentación en el conjunto de prueba, logramos aproximadamente un 98% de precisión. De hecho, fue similar a los resultados de la predicción cuando entrenamos el modelo. Estos resultados confirman que el modelo tiene

una fiabilidad más que aceptable. Por tanto, el modelo puede segmentar de forma autónoma cualquier imagen de satélite.

Las métricas del conjunto de datos que se muestran en la Tabla 3-3 proporcionan una descripción general de alto nivel del rendimiento de la red. Para ver el impacto que tiene cada clase en el rendimiento general, se inspecciona las métricas por clase.

Tabla 4-3: Métricas de evaluación por clase del modelo de segmentación semántica.

Clase	Precisión	Índice Jaccard	BFScore
'water'	0.96809	0.88053	0.76565
'ground'	0.97229	0.96373	0.87124

Realizado por: Nuñez, Christian, 2021.

La Tabla 4-3 nos muestra que el porcentaje de acierto alcanza un 96% cuando se segmenta la clase '*water'* y un 97% cuando se segmenta la clase '*ground'*, lo que nos hace entender que la red entrenada es levemente mejor segmentando la clase secundaria.

De las pruebas realizadas, podemos ver que la precisión promedio de la segmentación del recurso hídrico alcanzó el 97%. Esta precisión se mejora procesando posteriormente la máscara binaria generada aplicando PDI.

CONCLUSIONES

- Se implementó en un software con entorno gráfico, un algoritmo capaz de determinar el área de recursos hídricos mediante el uso de una red neuronal entrenada, y con un post procesado de la imagen en donde se le aplica procesamiento digital de imágenes.

- Se determinó que los métodos más comunes empleados en la Visión Artificial son: descomposición de matriz de color, procesos morfológicos, detección de bordes, y binarización de imágenes. Las Redes Neuronales aprenden mediante la experiencia, aprende de las entradas que le demos y los resultados que queremos obtener, reduciendo lo máximo posible el error.

- El método más común en la visión artificial para la detección de patrones es el *clustering*, que se emplea para el análisis de datos exploratorio con el fin de encontrar patrones o agrupaciones ocultos en los datos.

- Los datos compilados para el entrenamiento de la red neuronal se obtuvieron del sitio Sentinel Hub, el cual provee de imágenes satelitales capturadas por el satélite artificial Sentinel 2 en una gran variedad de espectros, siendo el infrarrojo uno de los más útiles para el estudio de recursos hídricos.

- Se realizó el entrenamiento de la red neuronal con un cantidad considerable de imágenes satelitales de recursos hídricos, lagos y lagunas más específicamente. Una vez obtenida la red, la misma fue capaz de etiquetar con un alta precisión el agua y la tierra firme.

- El algoritmo demostró ser altamente preciso a la hora de medir áreas de recursos hídricos, cabe recalcar que este algoritmo lo hace de manera autónoma y casi instantánea, en comparación con otros softwares que tienen el mismo propósito, los cuales son en la mayoría de los casos, manuales.

RECOMENDACIONES

- Se recomienda usar una versión del software Matlab lo más actual posible o que sea compatible con los toolbox de *Image Processing* y *Machine Learning*.

- Todas las imágenes a emplearse deben tener la misma resolución y orientación.

- Para acelerar la velocidad de entrenamiento de la red, se sugiere usar un computador que posea una GPU dedicada.

- Para obtener un mejor desempeño de la red neuronal, se recomienda entrenarla con imágenes en el espectro *False Color*, que hace un mejor contraste del agua con la tierra firme.

- Tener en cuenta la resolución espacial de las imágenes satelitales a la hora de descargarlas, ya que está se usa para el cálculo del área.

- No se recomienda trabajar con descomposición de matrices de color en el espectro RGB, ya que existen grandes diferencias de color entre recursos hídricos, lo que imposibilita obtener resultados exactos a la hora de binarizar las imágenes.

- Se recomienda transformar las imágenes de las cuales se vaya a calcular el área al espacio de color CIELAB.

- Hay que tener en cuenta la resolución de las imágenes de las cuales se vaya a realizar el cálculo del área, en el caso de ser de una resolución superior se deberá realizar un recorte de la misma resolución de entrada de la red neuronal.

BIBLIOGRAFÍA

AHUJA, S., PANIGRAHI, B.K., DEY, N., RAJINIKANTH, V. y GANDHI, T.K. "Deep transfer learning-based automated detection of COVID-19 from lung CT scan slices". *Applied Intelligence* [en línea], 2021, vol. 51, no. 1, pp. 571-585. ISSN 0924-669X. DOI 10.1007/s10489-020-01826-w. Disponible en: https://link.springer.com/10.1007/s10489-020-01826-w.

APPLE. *Performing Convolution Operations*. [blog]. [Consulta: 23 febrero 2021]. Disponible en:https://developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html.

ARANDA DELGADILLO, M., MEDINA MUÑOZ, L.A., RODRÍGUEZ ESPINOZA, I. y GONZÁLEZ LÓPEZ, S. "Aplicación de Técnicas de Visión Artificial y Sistemas Expertos para la Determinación del Valor de Monedas. *Revista de Prototipos Tecnológicos* [en línea], 2017, vol. 3, no. 7, pp. 7-12. Disponible en: https://www.ecorfan.org/spain/researchjournals/Prototipos_Tecnologicos/vol3num7/Revista_de_Prototipos_Tecnologicos_V3_N7_2.pdf

ARÉVALO ARÉVALO, W.D. Determinación de cambios de la cobertura arbórea usando imágenes satelitales Landsat 7ETM+A através de redes neuronales artificiales en la parroquia Achupallas, cantón Alausí, provincia de Chimborazo [en línea] (Trabajo de titulación). S.l.: Escuela Superior Politécnica de Chimborazo. 2016. Disponible en: http://dspace.espoch.edu.ec/handle/123456789/4874.

BASTIDA JUMILLA, M.C., MORALES SÁNCHEZ, J., MONEDERO, R.V., LARREY RUIZ, J. y SANCHO GÓMEZ, J.L. "Automatización de medidas morfológicas y ecogénicas de estructuras del aparato locomotor humano mediante procesado de imágenes ecográficas" [en línea], 2008. Disponible en: http://hdl.handle.net/10317/1366.

BERZAL, F. *Redes neuronales & Deep Learning*. Granada: s.n., 2018.

BONILLA CARRIÓN, C. Redes Convolucionales (Trabajo de titulación). S.l.: Universidad de Sevilla. 2020.

CARRILLO VELARDE, C.J. Segmentación semántica de imágenes para la obtención de rutas libres de obstáculos (Trabajo de titulación). S.l.: Universidad Internacional de La Rioja. 2021.

CASTELO CABAY, M.J., BUÑAY GUALOTO, G.I. y PILLAJO LANDA, B.G. Uso de Redes Neuronales Artificiales y Computación en la Nube para clasificar la cobertura del suelo en

territorio ecuatoriano, *Polo del Conocimiento* [en línea], 2021, vol. 6, pp. 14-28. Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=7927028.

CHICCHÓN APAZA, M.A. Fusión de Datos para Segmentación Semántica en Aplicaciones Urbanas de Teledetección Aérea Usando Algoritmos de Aprendizaje Profundo (Trabajo de titulación). S.l.: s.n. 2018.

COLQUE CAILLAHUA, P.E. Estimación de área glaciar utilizando Redes Neuronales convolucionales U-Net en imágenes multiespectrales sentinel 2 en el glaciar ausangate (Trabajo de titulación). S.l.: s.n. .2019.

COPERNICUS MASTERS. *SENTINEL HUB – A SATELLITE IMAGERY WEB SERVICE* [blog]. [Consulta: 13 febrero 2021]. Disponible en: https://copernicus-masters.com/winner/sentinel-hub-satellite-imagery-web-service/#.

COSMICUS, H. *El espectro electromagnético* [blog]. [Consulta: 13 febrero 2021]. Disponible en: https://lacienciaysusdemonios.com/2010/04/13/el-espectro-electromagnetico-i-introduccion/.

DARK, S. *Aprendizaje Automático: La Guía Definitiva para Principiantes para Comprender el Aprendizaje Automático*. S.l.: s.n. , 2018ª.

DARK, S. *Aprendizaje Profundo: Una Introducción a los Fundamentos del Aprendizaje Profundo Utilizando Python (Deep Learning Fundamentals Guide Spanish Edition / En Español)*. S.l.: s.n., 2018b.

DÍAZ PEDRAYES, Ó. Segmentación semántica de imágenes aéreas para la clasificación de cultivos (Trabajo de titulación). S.l.: s.n. 2021.

DUARTE, L.P., AGUILAR, L.J. y MÉNDEZ, S.P. "Mecanismos de articulación de infraestructuras de datos espaciales y programas de observación de la tierra [Spatial data infrastructure and earth observation joint mechanisms]", *Ventana Informática* [en línea], 2010, vol. 23. Disponible en: https://doi.org/10.30554/ventanainform.23.191.2010

ESPÍN VELASCO, L.I. Sistema inteligente de aprendizaje automático mediante el uso de un vehículo aéreo no tripulado (UAV) para el monitoreo de Oidium (sphaerotheca pannosa) en el cultivo de rosas de exportación en el Ecuador (Trabajo de titulación). S.l.: s.n. 2021.

FERREYRA, A. "LA OBSERVACIÓN DE LA TIERRA DESDE EL ESPACIO. IMÁGENES SATELITALES: UN RECURSO DISPONIBLE". *Instituto Nacional de Tecnología Agropecuaria*, 2016.

GARCIA GARCIA, A., ORTS ESCOLANO, S., OPREA, S., VILLENA MARTINEZ, V. y GARCIA RODRIGUEZ, J. "A review on deep learning techniques applied to semantic segmentation" [en línea], 2017. Disponible en: http://arxiv.org/abs/1704.06857.

GONZÁLES DIEZ, S. Evaluación del uso de redes neuronales convolucionales para clasificación de cultivos mediante imagen multiespectral (Trabajo de titulación). S.l.: s.n. 2019.

GONZÁLEZ GONZÁLEZ, M. "Diseño de un sistema de reconocimiento y detección automática de escritura en el aire mediante arquitecturas ResNet" [en línea], 2020. Disponible en: https://oa.upm.es/57860/.

HORMIGO RUIZ, D. Aplicación de Fully convolutional Neural Networks para segmentación semántica de imágenes para aplicaciones de conducción autónoma (Trabajo de titulación). S.l.: UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA. 2017.

INGEOEXPERT. *¿Qué es la teledetección y qué sistemas existen?* [blog]. [Consulta: 13 marzo 2021]. Disponible en: https://ingeoexpert.com/2018/07/06/que-es-la-teledeteccion/.

KONICA MINOLTA SENSING AMERICAS. *Entendiendo El Espacio de Color CIE L*A*B** [blog]. [Consulta: 17 enero 2021]. Disponible en: https://sensing.konicaminolta.us/mx/blog/entendiendo-el-espacio-de-color-cie-lab/.

MARCOS, A.G., DE PISÓN ASCACÍBAR, F.J.M., ELÍAS, F.A., LIMAS, M.C., MERÉ, J.B.O. y GONZÁLEZ, E.P.V. *Técnicas y algoritmos básicos de visión artificial*. S.l.: Universidad de la Rioja, 2006.

MATLAB. *MathWorks*. [blog]. [Consulta: 20 marzo 2021]. Disponible en: https://la.mathworks.com/.

MONASTERIO EXPÓSITO, L. Técnicas de segmentación semántica aplicadas en imágenes de laparoscopia (Trabajo de titulación). S.l.: Universidad de Alcalá Escuela Politécnica Superior. 2018.

MORALES, A. *El programa Copernicus: datos libres para la investigación* [blog]. [Consulta: 27 febrero 2021]. Disponible en: https://mappinggis.com/2020/02/el-programa-copernicus-datos-libres-para-la-investigacion/.

MOROCHO SANDE, K.L. y PULIG CESÉN, D.J. Análisis de Imágenes de Rayos X de Covid-19 a través de Redes Neuronales Artificiales (Trabajo de titulación). S.l.: Universidad de Guayaquil. 2021.

NA8. *¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador* [blog]. [Consulta: 06 febrero 2021]. Disponible en: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/.

NAPOLETANO, P., PICCOLI, F. y SCHETTINI, R. "Anomaly Detection in Nanofibrous Materials by CNN-Based Self-Similarity", *Sensors (Basel, Switzerland)* [en línea], 2018, vol. 18. Disponible en: http://www.ncbi.nlm.nih.gov/pubmed/29329268.

OROZCO, O.A., SAENZ, A.D., GONZÁLEZ, C.E. y QUINTANA, J.A. "Análisis del desempeño de redes neuronales profundas para segmentación semántica en hardware limitado". *ReCIBE, Revista electrónica de Computación, Informática, Biomédica y Electrónica* [en línea], 2019, vol. 8, no. 2. Disponible en: http://recibe.cucei.udg.mx/index.php/ReCIBE/article/view/142

PALMA MÉNDEZ, J. y MARÍN MORALES, R. *Inteligencia Artificial: Métodos, técnicas y aplicaciones*. Primera. Madrid: McGRAW-HILL, 2008.

PALOMINO, N.L.S. y CONCHA, U.R. "Técnicas de Segmentación en Procesamiento Digital de Imágenes". *Revista de investigación de Sistemas e Informática* [en línea], 2009, vol. 6. Disponible en: https://revistasinvestigacion.unmsm.edu.pe/index.php/sistem/article/view/3299

PÉREZ, A.B. Desarrollo de algoritmos de procesamiento de imagen avanzado para interpretación de imágenes médicas. Aplicación a segmentación de hígado sobre imágenes de Resonancia Magnética multisecuencia (Trabajo de titulación). Universidad de País Vasco (UPV/EHU). 2016.

PÉREZ, I. y GEGÚNDEZ, M. *Deep Learning*. S.l.: Universidad de Huelva, 2021.

PONCE CRUZ, P. *Inteligencia artificial con aplicaciones a la ingeniería*. S.l.: s.n., 2010.

PRIETO ORDAZ, O., RAMÍREZ ALONSO, G. y MALOOF FLORES, D. "Segmentación semántica para reconocimiento de escenas". *FINGUACH. Revista de Investigación Científica de la Facultad de Ingeniería de la Universidad Autónoma de Chihuahua* [en línea], 2019, vol. 6, pp. 6-7. Disponible en: https://vocero.uach.mx/index.php/finguach/article/view/371

PUSIOL, P. Redes Convolucionales en Comprension de Escenas (Trabajo de titulación). S.l.: Universidad Nacional de Córdoba. 2014.

PYTORCH TEAM. *Resnet* [blog]. [Consulta: 03 febrero 2021]. Disponible en: https://pytorch.org/hub/pytorch_vision_resnet/.

RODRÍGUEZ, D. *¿Cuál es la diferencia entre parámetro e hiperparámetro?* [blog]. [Consulta: 11 marzo 2021]. Disponible en: https://www.analyticslane.com/2019/12/16/cual-es-la-diferencia-entre-parametro-e-hiperparametro/.

ROMAN, E. y RUIZ, M. de F. Detección de macronutrientes y enfermedades en campos de cultivo de banano orgánico con Machine Learning (Trabajo de titulación). S.l.: Universidad de Piura. 2021

ROUHIAINEN, L. *Inteligencia artificial*. Madrid: Alienta Editorial, 2018.

SAEZ DE LA PASCUA, A. Deep learning para el reconocimiento facial de emociones básicas (Trabajo de titulación). S.l.: Universitat Politécnica de Catalunya. 2021.

SINERGISE LABORATORY FOR GEOGRAPHICAL INFORMATION SYSTEMS. *EO Browser* [blog]. [Consulta: 15 abril 2021]. Disponible en: https://www.space4water.org/softwaretoolweb-app/eo-browser.

SOLÁ, J. Identificación y estudio de masas de agua mediante el empleo de imágenes RADAR de Sentinel-1 (Trabajo de titulación). S.l.: Universitat Politècnica de València. 2021.

SUÁREZ, A.S., JIMÉNEZ, A., CASTRO FRANCO, M. y CRUZ ROA, A. "Clasificación y mapeo automático de coberturas del suelo en imágenes satelitales utilizando Redes Neuronales Convolucionales". *Orinoquia* [en línea], 2017, vol. 21, pp. 64-75. Disponible en: https://orinoquia.unillanos.edu.co/index.php/orinoquia/article/view/432

TINAJERO LEÓN, J.L., LOZADA YÁNEZ, P.E., ZABALA HARO, M.A. y JIMÉNEZ GRANIZO, C.N. "Sistema de Procesamiento Digital de Imágenes Satelitales para Cálculo de Áreas de Interés". *Ciencia Digital* [en línea], 2019, vol. 3, pp. 29-48. Disponible en: https://doi.org/10.33262/cienciadigital.v3i3.4..832

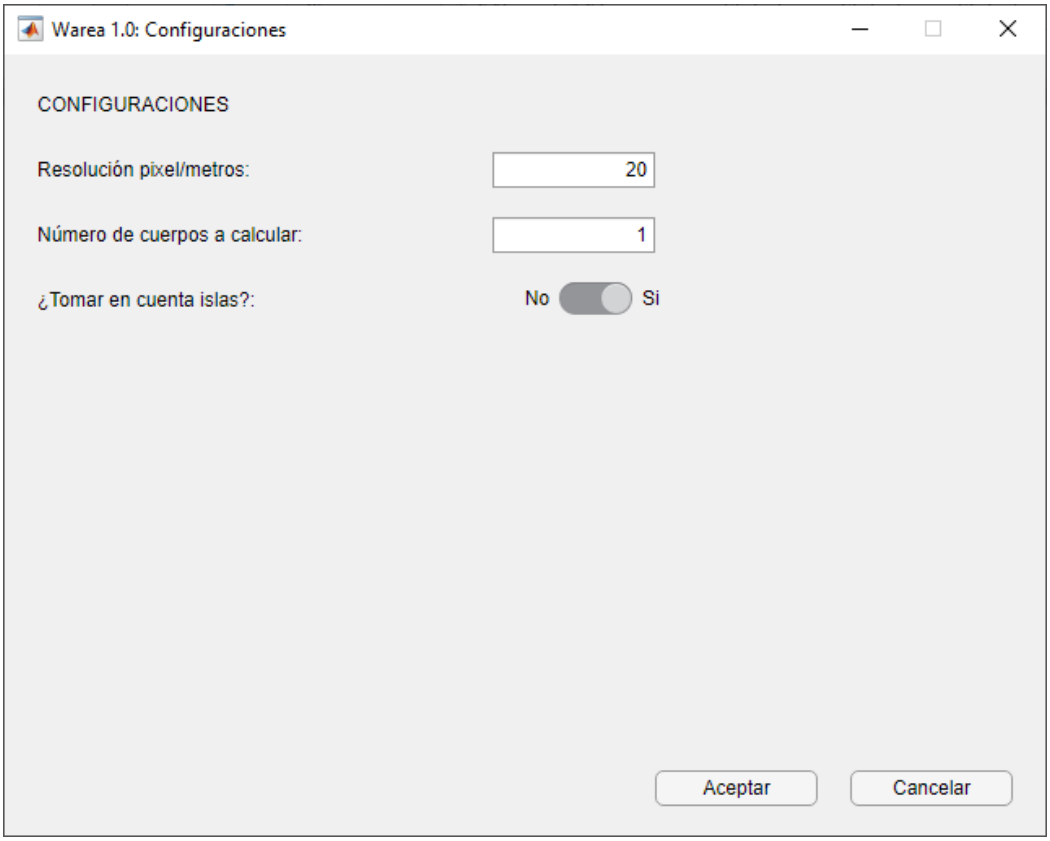
TITUAÑA JAMI, J.C. Desarrollo de un método para la clasificación automatizada de imágenes Landsat 8 mediante Redes Neuronales Artificiales (Trabajo de titulación). S.l.: s.n. 2018.

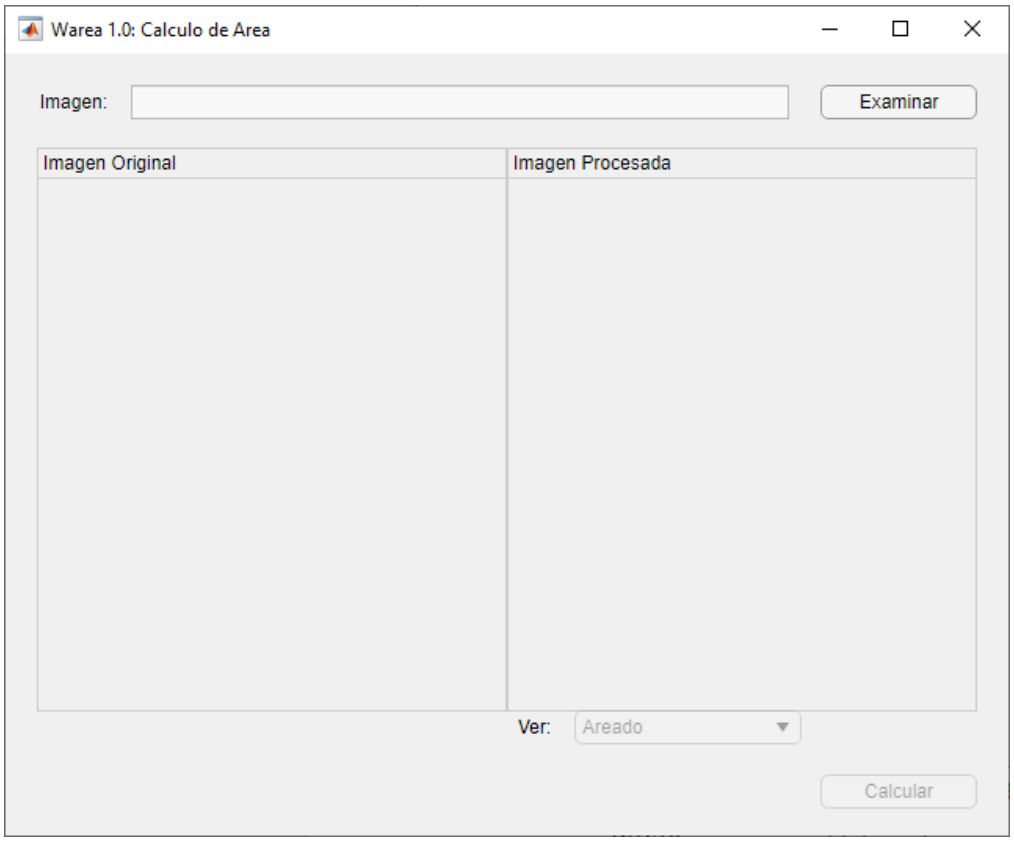
TORRES, A. "Procesamiento digital de imágenes". *Perfiles Educativos* [en línea], 1996, vol. 72. Disponible en: https://www.redalyc.org/articulo.oa?id=13207206

YU, D., WANG, H., CHEN, P. y WEI, Z. "Mixed Pooling for Convolutional Neural Networks", *Rough Sets and Knowledge Technology* [en línea], 2014. Disponible en: http://link.springer.com/10.1007/978-3-319-11740-9_34

ANEXOS

Anexo A. Imágenes del entorno gráfico del software.





Anexo B. Algoritmo de Pre-Procesamiento.

```
clc; clear; close all;

imds=imageDatastore('C:\Users\EQUIPO1\Desktop\Lagos\Lakes');

dim=length(imds.Files);

for i=1:dim
    I=readimage(imds,i);

    ISize=size(I);

    if ISize(2) > ISize(1)
        I=imrotate(I,90);
        ISize=size(I);
    end

    if ISize(1) > 960 && ISize(2) > 720
        targetSize = [960 720];
        r = centerCropWindow2d(size(I),targetSize);
        I = imcrop(I,r);
    end

    I=imresize(I,[960 720]);

        I_lab = rgb2lab(I);

        max_luminosity = 90;
        L = I_lab(:,:,1)/max_luminosity;
```

```
        I_adapthisteq = I_lab;
        I_adapthisteq(:,:,1) = adapthisteq(L)*max_luminosity;
        I_adapthisteq = lab2rgb(I_adapthisteq);

        imwrite(I_adapthisteq,"Lake_" + i + ".png")

    imwrite(I,"Lake_" + i + ".png")
end
```

Anexo C. Algoritmo de entrenamiento de red neuronal.

```
clc; close all; clear;

resnet18();

imgDir='C:\Users\EQUIPO1\Desktop\Lagos\Lakes Sized';
labelDir='C:\Users\EQUIPO1\Desktop\Lagos\Lakes
Labels\.LABELING_DEFINITIVE_54_SessionData';

imds=imageDatastore(imgDir);

lake=8;

I=readimage(imds, lake);

figure
imshow(I)

classes=["water", "ground"];
labelIDs=[1 2];

pxds=pixelLabelDatastore(labelDir,classes,labelIDs);

C=readimage(pxds,lake);
B=labeloverlay(I,C,'Colormap','spring','Transparency',0.4);
figure
imshow(B)

tbl=countEachLabel(pxds);

frequency = tbl.PixelCount/sum(tbl.PixelCount);

trainingImages = imds.Files(1:32);
valImages = imds.Files(33:43);
testImages = imds.Files(1:54);

imdsTrain = imageDatastore(trainingImages);
imdsVal = imageDatastore(valImages);
imdsTest = imageDatastore(testImages);

classes = pxds.ClassNames;

trainingLabels = pxds.Files(1:32);
valLabels = pxds.Files(33:43);
testLabels = pxds.Files(1:54);
```

```
pxdsTrain = pixelLabelDatastore(trainingLabels, classes, labelIDs);
pxdsVal = pixelLabelDatastore(valLabels, classes, labelIDs);
pxdsTest = pixelLabelDatastore(testLabels, classes, labelIDs);

% Especifique el tamaño de imagen de la red. Suele ser lo mismo que
los tamaños de las imágenes de entrenamiento.
imageSize = [960 720 3];

% Especifique el número de clases.
numClasses = numel(classes);

% Crear DeepLab v3+.
lgraph = deeplabv3plusLayers(imageSize, numClasses, "resnet18");

imageFreq = tbl.PixelCount ./ tbl.ImagePixelCount;
classWeights = median(imageFreq) ./ imageFreq;

pxLayer =
pixelClassificationLayer('Name','labels','Classes',tbl.Name,'ClassWeig
hts',classWeights);
lgraph = replaceLayer(lgraph,"classification",pxLayer);

% Definir datos de validación.
dsVal = combine(imdsVal,pxdsVal);

% Definir opciones de entrenamiento.
options = trainingOptions('sgdm', ...
    'LearnRateSchedule','piecewise',...
    'LearnRateDropPeriod',10,...
    'LearnRateDropFactor',0.3,...
    'Momentum',0.9, ...
    'InitialLearnRate',1e-3, ...
    'L2Regularization',0.005, ...
    'ValidationData',dsVal,...
    'MaxEpochs',30, ...
    'MiniBatchSize',8, ...
    'Shuffle','every-epoch', ...
    'CheckpointPath', tempdir, ...
    'VerboseFrequency',2,...
    'Plots','training-progress',...
    'ValidationPatience', 4);

dsTrain = combine(imds, pxds);

xTrans = [-10 10];
yTrans = [-10 10];
dsTrain = transform(dsTrain,
@(data)augmentImageAndLabel(data,xTrans,yTrans));

doTraining = false;
if doTraining
    [net, info] = trainNetwork(dsTrain,lgraph,options);
else
    data = load('trainedNetLakes54.mat');
    net = data.net;
end
```

```
C = semanticseg(I, net);

bw = C == 'water';
figure
imshow(bw)

B = labeloverlay(I,C,'Colormap',cmap,'Transparency',0.5);
figure
imshow(B)
pixelLabelColorbar(cmap, classes);

expectedResult = readimage(pxds,lake);
actual = uint8(C);
expected = uint8(expectedResult);
figure
imshowpair(actual, expected)

iou = jaccard(C,expectedResult);
table(classes,iou)

pxdsResults = semanticseg(imdsTest,net, ...
    'MiniBatchSize',4, ...
    'WriteLocation',tempdir, ...
    'Verbose',false);

metrics =
evaluateSemanticSegmentation(pxdsResults,pxdsTest,'Verbose',false);

metrics.DataSetMetrics

metrics.ClassMetrics

function data = augmentImageAndLabel(data, xTrans, yTrans)
% Aumenta las imágenes y las imágenes de etiquetas de píxeles
% mediante la reflexión y la traslación aleatorias.

for i = 1:size(data,1)

    tform = randomAffine2d(...
        'XReflection',true,...
        'XTranslation', xTrans, ...
        'YTranslation', yTrans);

    % Centre la vista en el centro de la imagen en el espacio de
salida mientras permite
    % que la traslación mueva la imagen de salida fuera de la vista.
    rout = affineOutputView(size(data{i,1}), tform, 'BoundsStyle',
'centerOutput');

    % Deforma las etiquetas de imagen y píxeles usando la misma
transformación.
    data{i,1} = imwarp(data{i,1}, tform, 'OutputView', rout);
    data{i,2} = imwarp(data{i,2}, tform, 'OutputView', rout);

end
end
```

```
function cmap = camvidColorMap()

cmap = [
    155 0 0   % Sky
    0 128 192       % Building
    ];

% Normalizar entre [0 1].
cmap = cmap ./ 255;
end

function pixelLabelColorbar(cmap, classNames)
% Agrega una barra de colores al eje actual. La barra de colores está
% formateada para mostrar los nombres de las clases con el color.

colormap(gca,cmap)

% Añade una barra de colores a la figura actual.
c = colorbar('peer', gca);

% Utilice los nombres de las clases para las marcas de graduación.
c.TickLabels = classNames;
numClasses = size(cmap,1);

% Centra la marca de verificación.
c.Ticks = 1/(numClasses*2):1/numClasses:1;

% Eliminar la marca de verificación.
c.TickLength = 0;
end
```

Anexo D. Código del software.

VENTANA DE INICIO

```
classdef Inicio < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        INICIO          matlab.ui.Figure
        Logo            matlab.ui.control.Image
        ComenzarButton  matlab.ui.control.Button
        ConfigsButton   matlab.ui.control.Button
        WAREA10Label    matlab.ui.control.Label
        Descripcion     matlab.ui.control.TextArea
    end


    properties (Access = private)
        ConfigsApp
        AreacionApp
        pixelmetros=20;
        numcuerpos=1;
        islas='Si';
    end

    methods (Access = public)
```

```
        function update(app, px, nc, is)
            app.pixelmetros = px;
            app.numcuerpos = nc;
            app.islas=is;
        end
    end


    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            app.pixelmetros;
            app.numcuerpos;
            app.islas;
        end

        % Button pushed function: ConfigsButton
        function ConfigsButtonPushed(app, event)
            app.ConfigsApp = Configs(app, app.pixelmetros,
app.numcuerpos, app.islas);
            app.ConfigsButton.Enable = 'off';
        end

        % Button pushed function: ComenzarButton
        function ComenzarButtonPushed(app, event)
            app.AreacionApp = Areacion(app, app.pixelmetros,
app.numcuerpos, app.islas);
            app.ComenzarButton.Enable = 'off';
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create INICIO and hide until all components are created
            app.INICIO = uifigure('Visible', 'off');
            app.INICIO.Position = [363 144 640 480];
            app.INICIO.Name = 'Warea 1.0: Inicio';
            app.INICIO.Resize = 'off';

            % Create Logo
            app.Logo = uiimage(app.INICIO);
            app.Logo.Position = [21 21 320 440];
            app.Logo.ImageSource = 'PicsArt_08-05-03.21.31.png';

            % Create ComenzarButton
            app.ComenzarButton = uibutton(app.INICIO, 'push');
            app.ComenzarButton.ButtonPushedFcn =
createCallbackFcn(app, @ComenzarButtonPushed, true);
            app.ComenzarButton.Position = [521 19 100 42];
            app.ComenzarButton.Text = 'Comenzar';

            % Create ConfigsButton
```

```
            app.ConfigsButton = uibutton(app.INICIO, 'push');
            app.ConfigsButton.ButtonPushedFcn = createCallbackFcn(app,
@ConfigsButtonPushed, true);
            app.ConfigsButton.Position = [521 81 100 40];
            app.ConfigsButton.Text = 'Configs';

            % Create WAREA10Label
            app.WAREA10Label = uilabel(app.INICIO);
            app.WAREA10Label.HorizontalAlignment = 'center';
            app.WAREA10Label.FontSize = 32;
            app.WAREA10Label.FontWeight = 'bold';
            app.WAREA10Label.Position = [381 421 240 40];
            app.WAREA10Label.Text = 'WAREA 1.0';

            % Create Descripcion
            app.Descripcion = uitextarea(app.INICIO);
            app.Descripcion.Editable = 'off';
            app.Descripcion.HorizontalAlignment = 'center';
            app.Descripcion.BackgroundColor = [0.9412 0.9412 0.9412];
            app.Descripcion.Position = [381 297 240 124];
            app.Descripcion.Value = {'Warea 1.0 es un software de
cálculo de áreas de recursos hídricos a travez de una red neuronal
convolucional y procesamiento digital de imagenes. El software permite
calcular el área de varios cuerpos a la vez, tomar o no en cuenta
islas y establecer la resolucion espacial de la imagen.'};

            % Show the figure after all components are created
            app.INICIO.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = Inicio

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.INICIO)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.INICIO)
        end
    end
```

```
end
```

VENTANA DE CONFIGURACIÓN

```
classdef Configs < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        CONFIGS                         matlab.ui.Figure
        CONFIGURACIONESLabel            matlab.ui.control.Label
        NmerodecuerposacalcularLabel    matlab.ui.control.Label
        TomarencuentaislasLabel         matlab.ui.control.Label
        ResolucinpixelmetrosLabel       matlab.ui.control.Label
        CancelarButton                  matlab.ui.control.Button
        AceptarButton                   matlab.ui.control.Button
        pixelmetros
matlab.ui.control.NumericEditField
        numcuerpos
matlab.ui.control.NumericEditField
        islas                           matlab.ui.control.Switch
    end

    properties (Access = private)
        CallingApp % Description
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app, mainapp, pixelmetros, numcuerpos,
islas)
            app.CallingApp = mainapp;

            app.pixelmetros.Value=pixelmetros;
            app.numcuerpos.Value=numcuerpos;
            app.islas.Value=islas;
        end

        % Close request function: CONFIGS
        function CONFIGSCloseRequest(app, event)
            app.CallingApp.ConfigsButton.Enable = 'on';
            delete(app)

        end

        % Button pushed function: CancelarButton
        function CancelarButtonPushed(app, event)
            app.CallingApp.ConfigsButton.Enable = 'on';
            delete(app)

        end

        % Button pushed function: AceptarButton
        function AceptarButtonPushed(app, event)
            update(app.CallingApp, app.pixelmetros.Value,
app.numcuerpos.Value, app.islas.Value);
            app.CallingApp.ConfigsButton.Enable = 'on';
```

```
            delete(app)
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create CONFIGS and hide until all components are created
            app.CONFIGS = uifigure('Visible', 'off');
            app.CONFIGS.Position = [363 144 640 480];
            app.CONFIGS.Name = 'Warea 1.0: Configuraciones';
            app.CONFIGS.Resize = 'off';
            app.CONFIGS.CloseRequestFcn = createCallbackFcn(app,
@CONFIGSCloseRequest, true);

            % Create CONFIGURACIONESLabel
            app.CONFIGURACIONESLabel = uilabel(app.CONFIGS);
            app.CONFIGURACIONESLabel.Position = [21 439 124 22];
            app.CONFIGURACIONESLabel.Text = 'CONFIGURACIONES';

            % Create NmerodecuerposacalcularLabel
            app.NmerodecuerposacalcularLabel = uilabel(app.CONFIGS);
            app.NmerodecuerposacalcularLabel.Position = [21 359 169
22];
            app.NmerodecuerposacalcularLabel.Text = 'Número de cuerpos
a calcular:';

            % Create TomarencuentaislasLabel
            app.TomarencuentaislasLabel = uilabel(app.CONFIGS);
            app.TomarencuentaislasLabel.Position = [21 319 140 22];
            app.TomarencuentaislasLabel.Text = '¿Tomar en cuenta
islas?:';

            % Create ResolucinpixelmetrosLabel
            app.ResolucinpixelmetrosLabel = uilabel(app.CONFIGS);
            app.ResolucinpixelmetrosLabel.Position = [21 399 136 22];
            app.ResolucinpixelmetrosLabel.Text = 'Resolución
pixel/metros:';

            % Create CancelarButton
            app.CancelarButton = uibutton(app.CONFIGS, 'push');
            app.CancelarButton.ButtonPushedFcn =
createCallbackFcn(app, @CancelarButtonPushed, true);
            app.CancelarButton.Position = [521 19 100 22];
            app.CancelarButton.Text = 'Cancelar';

            % Create AceptarButton
            app.AceptarButton = uibutton(app.CONFIGS, 'push');
            app.AceptarButton.ButtonPushedFcn = createCallbackFcn(app,
@AceptarButtonPushed, true);
            app.AceptarButton.Position = [401 19 100 22];
            app.AceptarButton.Text = 'Aceptar';

            % Create pixelmetros
            app.pixelmetros = uieditfield(app.CONFIGS, 'numeric');
```

```
            app.pixelmetros.Limits = [1 Inf];
            app.pixelmetros.Position = [301 399 100 22];
            app.pixelmetros.Value = 1;

            % Create numcuerpos
            app.numcuerpos = uieditfield(app.CONFIGS, 'numeric');
            app.numcuerpos.Limits = [1 10];
            app.numcuerpos.Position = [301 359 100 22];
            app.numcuerpos.Value = 1;

            % Create islas
            app.islas = uiswitch(app.CONFIGS, 'slider');
            app.islas.Items = {'No', 'Si'};
            app.islas.Position = [342 321 45 20];
            app.islas.Value = 'Si';

            % Show the figure after all components are created
            app.CONFIGS.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = Configs(varargin)

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.CONFIGS)

            % Execute the startup function
            runStartupFcn(app, @(app)startupFcn(app, varargin{:}))

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.CONFIGS)
        end
    end
end
```

VENTANA DE CALCULO DE ÁREA

```
classdef Areacion < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        AREACION              matlab.ui.Figure
        ImagenLabel           matlab.ui.control.Label
```

```
        ImagenEditField       matlab.ui.control.EditField
        ExaminarButton        matlab.ui.control.Button
        CalcularButton        matlab.ui.control.Button
        ImagenOriginalPanel   matlab.ui.container.Panel
        Original              matlab.ui.control.UIAxes
        ImagenProcesadaPanel  matlab.ui.container.Panel
        Areado                matlab.ui.control.UIAxes
        VerDropDownLabel      matlab.ui.control.Label
        VerDropDown           matlab.ui.control.DropDown
    end

    properties (Access = private)
        CallingApp
        pixelmetros;
        numcuerpos;
        islas;
    end

    methods (Access = private)

        %Ajuste de imagen
        function Imagen = imgrr(app, I)

            ISize=size(I);

            if ISize(2) > ISize(1)
                I=imrotate(I,90);
                ISize=size(I);
            end

            if ISize(1) > 960 || ISize(2) > 720
                targetSize = [960 720];
                r = centerCropWindow2d(size(I),targetSize);
                I = imcrop(I,r);
            end

            Imagen=imresize(I,[960 720]);
        end

        %Segmentacion CNN
        function [seg, masc] = imgseg(app, I, net, numc)

            seg = semanticseg(I, net);
            bw = seg == 'water';
            masc = bwareafilt(bw,numc);

        end

        %Operaciones morfologicas
        function BW = imgmorf(app, I, masc, numc, islas)

            % Erode mask with disk
            radius = 10;
            decomposition = 0;
            se = strel('disk', radius, decomposition);
            BW = imerode(masc, se);

            % Open mask with disk
```

```
            radius = 10;
            decomposition = 0;
            se = strel('disk', radius, decomposition);
            BW = imopen(BW, se);

            Ilab = rgb2lab(I);

            % Active contour
            iterations = 500;
            BW = activecontour(Ilab, BW, iterations, 'Chan-Vese',
'ContractionBias',0.3);

            % Erode mask with disk 2
            radius = 10;
            decomposition = 0;
            se = strel('disk', radius, decomposition);
            BW = imerode(BW, se);

            % Active contour 2
            iterations = 100;
            BW = activecontour(Ilab, BW, iterations, 'Chan-Vese');

            BW = bwareafilt(BW, numc);

            if islas == 'No'
                BW = imfill(BW, 'holes');
            end

        end

        %Areado
        function area = imgarea(app, BW, pxm)
            [B,L] = bwboundaries(BW,'noholes');
            h=figure;
            h.Visible='off';
            imshow(label2rgb(L, @jet, [.5 .5 .5]))
            hold on

            for k = 1:length(B)
                boundary = B{k};
                plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth',
2)
            end

            stats = regionprops(L,'Area','Centroid');
            threshold = 0.94;
            for k = 1:length(B)
                area = (((pxm/1000)^2)*stats(k).Area);
                area_string = sprintf('%2.2f',area);
                if area > threshold
                    centroid = stats(k).Centroid;
                    plot(centroid(1),centroid(2),'ko');
                end
                text(boundary(1,2)-35,boundary(1,1)+13,area_string +
"km2",'Color','k',...
                    'FontSize',14,'FontWeight','bold');
            end
```

```
            savefig('Areado.fig');
            close

        end

    end


    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app, mainapp, pixelmetros, numcuerpos,
islas)
            app.CalcularButton.Enable='off';
            app.VerDropDown.Enable='off';

            app.Original.Visible='off';
            app.Areado.Visible='off';

            app.CallingApp = mainapp;

            app.pixelmetros = pixelmetros;
            app.numcuerpos = numcuerpos;
            app.islas=islas;

            resnet18();
        end

        % Close request function: AREACION
        function AREACIONCloseRequest(app, event)
            app.CallingApp.ComenzarButton.Enable = 'on';
            delete(app)
        end

        % Button pushed function: ExaminarButton
        function ExaminarButtonPushed(app, event)
            global I
            [file, path]=uigetfile({'*.jpg; *.jpeg; *.png','Imágenes';
'*.*' ,'Todos los archivos'},'Seleccione una imagen');
            imdireccion=strcat(path, file);
            I=imread(imdireccion);
            app.ImagenEditField.Value=imdireccion;

            %Redimensionamiento y rotación de imagen
            I=imgrr(app, I);

            app.Original.Visible='on';
            imshow(I,'Parent',app.Original)

            app.CalcularButton.Enable='on';
        end

        % Button pushed function: CalcularButton
        function CalcularButtonPushed(app, event)
            app.CalcularButton.Enable='off';
            app.ExaminarButton.Enable='off';
```

```
        app.VerDropDown.Enable='off';

        global I seg BW

        imshow(I,'Parent',app.Areado)

        %Segmentación por medio de CNN
        load('trainedNetLakes24062021.mat');

        [seg, masc] = imgseg(app, I, net, app.numcuerpos);

        app.Areado.Visible='on';
        imshow(labeloverlay(I,seg),'Parent', app.Areado)
        pause(0.5)

        %Operaciones morfologicas
        BW = imgmorf(app, I, masc, app.numcuerpos, app.islas);


        imshow(labeloverlay(I,BW,'Colormap','spring','Transparency'
        ,0.75),'Parent',app.Areado);
        pause(0.5)

        %Areado
        imgarea(app, BW, app.pixelmetros);

        figHandle = openfig('Areado.fig');
        axHandle = gca(figHandle);
        copyobj(axHandle.Children, app.Areado)

        app.CalcularButton.Enable='on';
        app.ExaminarButton.Enable='on';

        app.VerDropDown.Value='Areado';
        app.VerDropDown.Enable='on';

    end

    % Value changed function: VerDropDown
    function VerDropDownValueChanged(app, event)
        global I seg BW

        value = app.VerDropDown.Value;

        switch value
            case 'Segmentacion'
                imshow(labeloverlay(I,seg),'Parent', app.Areado)
            case 'Morfologicas'

                    imshow(labeloverlay(I,BW,'Colormap','spring','T
                    ransparency',0.75),'Parent',app.Areado);
            case 'Areado'
                figHandle = openfig('Areado.fig');
                axHandle = gca(figHandle);
                copyobj(axHandle.Children, app.Areado)
        end
    end
```

```
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create AREACION and hide until all components are
created
            app.AREACION = uifigure('Visible', 'off');
            app.AREACION.Position = [100 100 640 499];
            app.AREACION.Name = 'Warea 1.0: Calculo de Area';
            app.AREACION.CloseRequestFcn = createCallbackFcn(app,
@AREACIONCloseRequest, true);

            % Create ImagenLabel
            app.ImagenLabel = uilabel(app.AREACION);
            app.ImagenLabel.HorizontalAlignment = 'right';
            app.ImagenLabel.Position = [17 458 49 22];
            app.ImagenLabel.Text = 'Imagen:';

            % Create ImagenEditField
            app.ImagenEditField = uieditfield(app.AREACION, 'text');
            app.ImagenEditField.Editable = 'off';
            app.ImagenEditField.Position = [81 458 420 22];

            % Create ExaminarButton
            app.ExaminarButton = uibutton(app.AREACION, 'push');
            app.ExaminarButton.ButtonPushedFcn =
createCallbackFcn(app, @ExaminarButtonPushed, true);
            app.ExaminarButton.Position = [521 458 100 22];
            app.ExaminarButton.Text = 'Examinar';

            % Create CalcularButton
            app.CalcularButton = uibutton(app.AREACION, 'push');
            app.CalcularButton.ButtonPushedFcn =
createCallbackFcn(app, @CalcularButtonPushed, true);
            app.CalcularButton.Position = [521 18 100 22];
            app.CalcularButton.Text = 'Calcular';

            % Create ImagenOriginalPanel
            app.ImagenOriginalPanel = uipanel(app.AREACION);
            app.ImagenOriginalPanel.Title = 'Imagen Original';
            app.ImagenOriginalPanel.Position = [21 80 300 360];

            % Create Original
            app.Original = uiaxes(app.ImagenOriginalPanel);
            title(app.Original, '')
            xlabel(app.Original, '')
            ylabel(app.Original, '')
            app.Original.Position = [1 0 300 340];

            % Create ImagenProcesadaPanel
            app.ImagenProcesadaPanel = uipanel(app.AREACION);
            app.ImagenProcesadaPanel.Title = 'Imagen Procesada';
            app.ImagenProcesadaPanel.Position = [321 80 300 360];
```

```
            % Create Areado
            app.Areado = uiaxes(app.ImagenProcesadaPanel);
            title(app.Areado, '')
            xlabel(app.Areado, '')
            ylabel(app.Areado, '')
            app.Areado.Position = [1 0 300 340];

            % Create VerDropDownLabel
            app.VerDropDownLabel = uilabel(app.AREACION);
            app.VerDropDownLabel.HorizontalAlignment = 'right';
            app.VerDropDownLabel.Position = [322 59 27 22];
            app.VerDropDownLabel.Text = 'Ver:';

            % Create VerDropDown
            app.VerDropDown = uidropdown(app.AREACION);
            app.VerDropDown.Items = {'Segmentacion', 'Morfologicas',
'Areado'};
            app.VerDropDown.ValueChangedFcn = createCallbackFcn(app,
@VerDropDownValueChanged, true);
            app.VerDropDown.Position = [364 59 145 22];
            app.VerDropDown.Value = 'Areado';

            % Show the figure after all components are created
            app.AREACION.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = Areacion(varargin)

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.AREACION)

            % Execute the startup function
            runStartupFcn(app, @(app)startupFcn(app, varargin{:}))

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.AREACION)
        end
    end
end
```

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

DIRECCIÓN DE BIBLIOTECAS Y RECURSOS DEL APRENDIZAJE

UNIDAD DE PROCESOS TÉCNICOS

REVISIÓN DE NORMAS TÉCNICAS, RESUMEN Y BIBLIOGRAFÍA

Fecha de entrega: 10 / 02 / 2022

INFORMACIÓN DEL AUTOR
Nombres – Apellidos: CHRISTIAN ALEXANDER NÚÑEZ SEGOVIA
INFORMACIÓN INSTITUCIONAL
Facultad: INFORMÁTICA Y ELECTRÓNICA
Carrera: ELECTRÓNICA Y AUTOMATIZACIÓN
Título a optar: INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN
f. Analista de Biblioteca responsable: Firmado electrónicamente por: ELIZABETH FERNANDA AREVALO MEDINA