Revista GEOESPACIAL (Julio-Diciembre 2023) 20(2): 01-19
ISSN: 2600-5921

NUEVAS TECNOLOGÍAS DE APRENDIZAJE PROFUNDO PARA EL PROCESAMIENTO DE DATOS GEOESPACIALES

NEW DEEP LEARNING TECHNOLOGIES FOR GEOSPATIAL DATA PROCESSING

Luiz Claudio Oliveira de Andrade [1*], Maurício Carvalho Mathias de Paulo [2]

[1] Departamento de Ciencias de la Tierra y Construcción de la Universidad de las Fuerzas Armadas ESPE, Av. Gral. Rumiñahui S/N, Sangolquí, 171103, Ecuador, luizclaudio.andrade@eb.mil.br
[2] Programa de Postgrado en Ingeniería de Defensa del Instituto Militar de Ingeniería, Praça Gen. Tibúrcio, 80 - Urca, Rio de Janeiro - RJ, 22290-270, Brasil, mauricio.paulo@ime.eb.br
** Autor de correspondencia: Av. Gral. Rumiñahui S/N, Sangolquí, 171103, Ecuador y luizclaudio.andrade@eb.mil.br.*

Recibido: 03 de octubre de 2023 / Aceptado: 01 de diciembre de 2023

RESUMEN

El uso de la Inteligencia Artificial ha impactado de manera innegable en la forma en que se procesan los datos en la actualidad. Las redes neuronales, en constante evolución, están logrando resultados cada vez mejores en diversas tareas. En este contexto, cabe mencionar actividades como la clasificación de imágenes, la detección de objetos y la segmentación semántica de imágenes, las cuales tienen un valor fundamental en el procesamiento de información geoespacial. Sin embargo, las tecnologías de visión computacional ampliamente conocidas, como PyTorch y TensorFlow, se enfrentan a desafíos al abordar la capa adicional de complejidad presentada por los datos espaciales. Estos datos se caracterizan, entre otros aspectos, por el gran volumen de datos por archivo, la presencia de diversas bandas espectrales y diferentes sistemas de referencia espacial. Por lo tanto, han surgido varias iniciativas encaminadas a adaptar las tecnologías de aprendizaje profundo para manejar datos espaciales. En este contexto, este trabajo tiene como objetivo presentar y analizar dos bibliotecas innovadoras: TorchGeo y RasterVision. Ambas fueron diseñadas con el propósito de manejar las características especiales de los datos geoespaciales. A lo largo del desarrollo de este estudio, se han implementado códigos para segmentar imágenes de satélites utilizando ambas bibliotecas. El análisis resultante corrobora que ambas pueden ser utilizadas de manera eficiente en el manejo de datos geoespaciales. No obstante, es crucial señalar que cada una posee características distintas, las cuales son comparadas para permitir a los profesionales en el campo de la geoinformación tomar decisiones informadas sobre cuál de las dos utilizar en sus tareas de procesamiento de datos geoespaciales.

Palabras clave: geoprocesamiento; aprendizaje profundo; redes neuronales; inteligencia artificial

ABSTRACT

The use of Artificial Intelligence has undeniably impacted the way data is processed today. Neural networks, constantly evolving, are achieving increasingly better results in various tasks. In this context, it is worth mentioning activities such as image classification, object detection, and semantic image

segmentation, which play a fundamental role in geospatial information processing. However, widely known computer vision technologies like PyTorch and TensorFlow face challenges when addressing the additional layer of complexity presented by spatial data. These data are characterized, among other aspects, by the large volume of data per file, the presence of various spectral bands, and different spatial reference systems. Therefore, several initiatives have emerged to adapt deep learning technologies to handle spatial data. In this context, this work aims to present and analyze two innovative libraries: TorchGeo and RasterVision. Both were designed with the purpose of handling the special characteristics of geospatial data. Throughout the development of this study, code has been implemented to segment satellite images using both libraries. The resulting analysis confirms that both can be efficiently used in handling geospatial data. However, it is crucial to note that each one possesses distinct features, which are compared to enable professionals in the field of geoinformation to make informed decisions about which one to use in their geospatial data processing tasks.

Keywords: geoprocessing; deep learning; neural networks; artificial intelligence

INTRODUCCIÓN

La información geoespacial (geoinformación) (Konecny, 2014), tiene gran valor en el proceso de toma de decisiones de distintas áreas. Su uso es fundamental en áreas como Operaciones Militares, Dinámica Regional, Seguridad Pública, Ordenamiento Territorial e incluso en el área comercial.

Hoy en día, el uso de geoinformación se impulsa por la enorme cantidad de datos generados diariamente por diversos tipos de sensores. Sin embargo, la gran cantidad de datos producida con una velocidad cada vez mayor exige una mayor capacidad de procesamiento. En ese contexto, se puede ver en la actualidad varias iniciativas enfocadas en el uso de técnicas de aprendizaje profundo (*deep learning*) (Lecun, Bengio and Hinton, 2015) para el procesamiento de geoinformación, lo que ha permitido obtener diversos avances en el área de la visión computacional (*computer vision*) (O'Mahony *et al.*, 2020).

Actividades como Clasificación de Imágenes, Detección de Objetos y Segmentación Semántica de Imágenes realizadas con técnicas de aprendizaje profundo logran avances día a día que impactan de manera innegable la manera como se procesa geoinformación hoy día. No obstante, el procesamiento de geoinformación sigue siendo un desafío para las técnicas de aprendizaje profundo debido a su capa extra de complejidad. Complejidad que es caracterizada, entre otras, por el gran volumen de datos por archivo, la presencia de diversas bandas espectrales y diferentes sistemas de referencia espacial.

Distintas librerías de visión computacional ampliamente conocidas, como PyTorch (Ketkar *et al.*, 2021) y TensorFlow (Dillon *et al.*, 2017), no tienen la capacidad nativa de trabajar con datos geoespaciales. Su uso para el procesamiento de eses tipos de datos requiere una codificación adicional para que las características espaciales de eses datos no se pierdan. En este contexto, este trabajo tiene como objetivo presentar y analizar dos bibliotecas innovadoras: Torchgeo (Stewart *et al.*, 2021) y RasterVision (Nachmany and Alemohammad, 2019). Ambas fueron diseñadas con el propósito de manejar las características especiales de los datos geoespaciales.

A lo largo del desarrollo de este estudio, se han implementado códigos para llevar a cabo tareas de segmentación semántica de imágenes de satélites utilizando ambas bibliotecas. Las dos bibliotecas fueron analizadas cuánto a la carga de datos, configuración del entrenamiento, entrenamiento y capacidad de realizar una predicción georreferenciada.

La contribución de este estudio radica en proporcionar información que permita a los profesionales en el campo de la geoinformación tomar decisiones informadas sobre cuál de las dos utilizar en sus tareas de procesamiento de datos geoespaciales.

TECNOLOGÍAS DE APRENDIZAJE PROFUNDO PARA EL PROCESAMIENTO DE DATOS GEOSPACIALES

Por lo general, datos de imágenes de satélite son suministrados en formatos de escenas. Estas escenas son comprendidas como tensores $I \in \mathbb{R}^{HxWxC}$, donde H es altura, W es ancho y C es número de canales espectrales (Stewart *et al.*, 2022). Asimismo, estos datos van acompañados de metadatos (Quarati, De Martino and Rosim, 2021), que contienen informaciones sobre su Sistema de Referencia Espacial (CRS), resolución espacial, límites espaciales (bbox) e informaciones temporales (es decir, el *timestamp*).

Trabajar con datos espaciales implica la capacidad de trabajar con todas sus características especiales. Es decir, se debe poder trabajar con datos en diferentes sistemas de referencia espacial, diferentes bandas espectrales, diferentes resoluciones espaciales, diferentes límites espaciales y con diferentes informaciones temporales.

En general, la combinación de diferentes datos geoespaciales se puede realizar utilizando otras herramientas geoespaciales, como por ejemplo QGIS (Congedo, 2021). Estas herramientas permiten combinar los datos mediante conversiones de CRS, resolución espacial, y recortes que permitan el alineamiento de los límites espaciales (Figura 1). De esa forma, es posible procesar datos espaciales con bibliotecas de aprendizaje profundo como PyTorch.

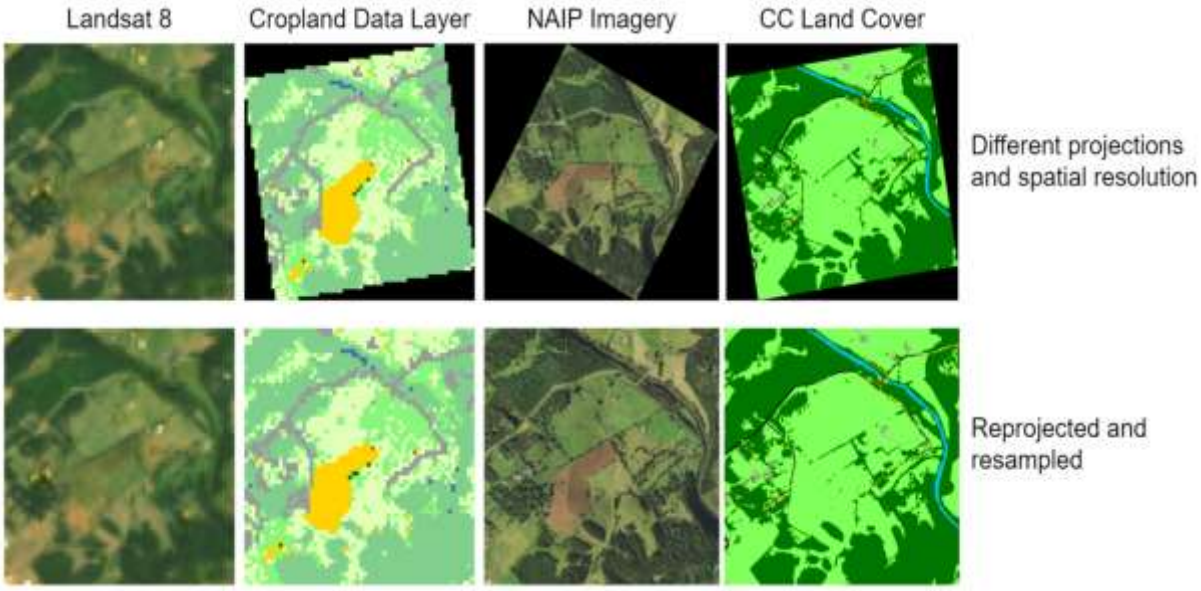

Figura 1 - El desafío de lidiar con datos geoespaciales (Stewart *et al.*, 2022)

Sin embargo, además de generar duplicación de datos, ese enfoque requiere que los profesionales de geoprocesamiento desarrollen maneras de mantener los metadatos (sobre todo bbox y CRS) para que las inferencias (Stevens, Antiga and Viehmann, 2020) puedan ser georreferenciadas. Si no hay ese control, toda la información geoespacial se pierde.

En este contexto, se presentan TorchGeo y RasterVision. Ambas bibliotecas facilitan el procesamiento de datos geoespaciales, permitiendo su aplicación en tareas de visión computacional y generando inferencias georreferenciadas. A continuación, se abordarán ambas en mayor detalle.

TORCHGEO

TorchGeo es una biblioteca de Python (Borges, 2014) que permite el procesamiento de datos geoespaciales en el ecosistema de aprendizaje profundo de PyTorch. TorchGeo es la primera biblioteca que proporciona modelos previamente entrenados para imágenes satelitales multiespectrales, lo que permite el aprendizaje por transferencia en tareas de teledetección (Stewart *et al.*, 2022).

TorchGeo permite el procesamiento de datos geoespaciales a través de sus submódulos, que incluyen:

1. *Datasets*: Proveen acceso a datos geoespaciales de referencia (*benchmark datasets*), como Landsat y Sentinel. Permiten la creación de conjuntos de datos personalizados y ofrecen operadores espaciales para realizar unión e intersección de *datasets*;
2. *Sampler*: Son muestreadores que permiten extraer muestras de los conjuntos de datos. Utilizan límites espaciales (bbox) que pueden generarse de manera aleatoria o sistemática;
3. *Models*: Son implementaciones de arquitecturas de modelos de aprendizaje profundo comunes con entradas de tamaño variable y pesos previamente entrenados. Esto permite la utilización de todas las bandas multiespectrales de las imágenes utilizadas;
4. *Transformers*: Son aumentos (*augmentations*) que permiten la utilización en imágenes con varias bandas espectrales, no solamente RGB; y
5. *Trainers*: Son entrenadores basados en la biblioteca Pytorch Lightning (Borovec *et al.*, 2022) que permiten entrenar los modelos de TorchGeo.

RASTERVISION

RasterVision es una biblioteca de Python que conecta el mundo de los Sistemas de Informaciones Geográficas (SIG) con el mundo de la visión computacional basada en el aprendizaje profundo. Proporciona un flujo configurable (Figura 2) de procesamiento de datos geoespaciales que funciona en tareas de clasificación de imágenes, segmentación semántica y detección de objetos (Azavea, 2022).

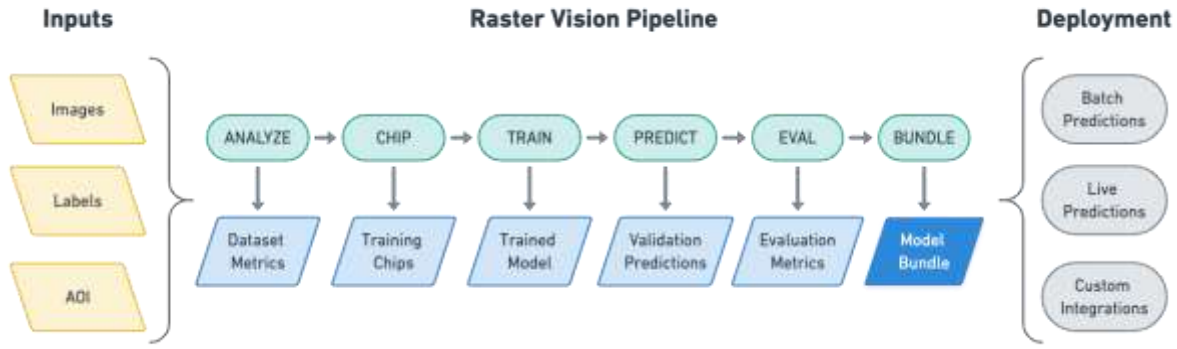

Figura 2 - Flujo configurable de RasterVision (Azavea, 2022)

RasterVision divide el procesamiento de datos geoespaciales en cuatro pasos, a saber:

1. Lectura de datos geoespaciales: este paso es responsable por generar los conjuntos de datos (los *GeoDataset*) que serán utilizados. Los conjuntos de datos pueden proporcionar los datos de manera aleatoria o sistemática. Asimismo, pueden ser definidos aumentos (*augmentations*);
2. Entrenamiento del modelo: paso que cría un *Learner* a partir de configuraciones que definen, entre otros aspectos, los datos y los modelos que serán utilizados para el entrenamiento;

3. Realizar predicciones: utilizando un *Learner* ya entrenado y un *GeoDataset* sistemático, es posible generar las predicciones; y
4. Guardar las predicciones: las predicciones generadas pueden ser guardadas como datos geoespaciales de manera transparente para el usuario.

MATERIALES Y MÉTODOS

Para realizar un análisis más detallado de las dos bibliotecas, se decidió llevar a cabo una tarea de segmentación semántica. Por lo tanto, se crearon dos cuadernos en Kaggle (Bojer and Meldgaard, 2020), un utilizando TorchGeo 0.4.1 (Andrade, 2023b) y otro utilizando RasterVision 0.21 (Andrade, 2023a). El conjunto de datos utilizado para el entrenamiento, validación y pruebas es lo mismo para los dos cuadernos.

El flujo de trabajo general utilizado en este estudio contiene 4 pasos, los cuales están presentes en la Figura 3 y serán analizados en detalles en las siguientes subsecciones.

Figura 3 - Flujo de Trabajo Ejecutado

CONJUNTO DE DATOS UTILIZADO

El conjunto de datos utilizado fue elaborado con 4 imágenes obtenidas del proyecto NAIP (National Aeronautics and Space Administration, 2020) y 4 etiquetas obtenidas de Chesapeake (Chesapeake Bay Program Office (CBPO), 2022) (Figura 4). Ambos conjuntos de datos están en el CRS NAD83 / UTM zona 18N (EPSG 26918). Dado que el conjunto de datos de Chesapeake abarca un área más grande que las imágenes NAIP utilizadas, fueron generados recortes de los datos de Chesapeake para generar 4 etiquetas que coincidieran con las extensiones de las imágenes NAIP. Además, se ajustaron los tamaños de píxeles para que fueran iguales (0.6 m). También se generaron archivos geojson (EPSG 4326) para las extensiones de las imágenes NAIP, que pueden usarse como áreas de interés (AOI).

Es importante señalar que las imágenes fueron obtenidas por sensores aerotransportados y poseen 4 bandas espectrales (R-G-B-NIR).

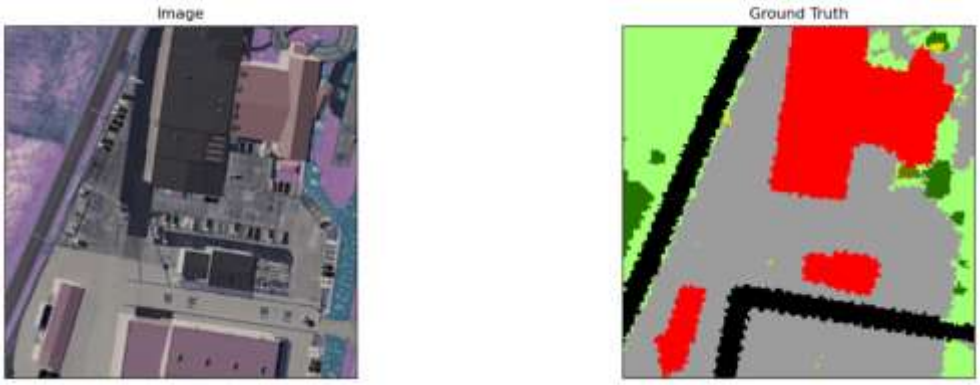

Figura 4 - Recorte del conjunto de datos utilizado

Las etiquetas son separadas en 13 clases más valores de fondo, a saber (Chesapeake Bay Program Office (CBPO), 2022):

1. Agua: Todas las áreas de aguas abiertas, incluyendo ríos, lagos y embarcaciones no amarradas a muelles. También abarca pequeños elementos antropogénicos como estanques agrícolas y estructuras de retención de aguas pluviales.
2. Humedales: Áreas de vegetación baja ubicadas a lo largo de regiones marinas o estuarinas que visualmente presentan el aspecto de suelo saturado que rodea la vegetación, ubicadas a lo largo de importantes vías fluviales.
3. Dosel de Árboles: Vegetación leñosa de hoja caduca y siempre verde, ya sea de sucesión natural o de plantación humana, que mide más de aproximadamente 3 metros de altura.
4. Arbusto: Área heterogénea de vegetación leñosa tanto caducifolia como perenne. Caracterizada por la variación en la altura de la vegetación con una cobertura irregular de arbustos y árboles jóvenes intercalados con pastos y otra vegetación más baja.
5. Vegetación Baja: Material vegetal de menos de 3 metros de altura aproximadamente. Incluye césped, campos labrados, plantaciones de viveros con o sin cubierta de lona, áreas de manejo forestal recientemente taladas y cobertura natural del suelo.
6. Árido: Áreas desprovistas de vegetación consistente en material terrestre natural, independientemente de cómo haya sido talado. Esto incluye playas, marismas y terrenos desnudos en sitios de construcción.
7. Estructuras: Objetos construidos por humanos hechos de materiales impermeables que tienen más de aproximadamente 2 metros de altura, como casas, centros comerciales y torres eléctricas.
8. Superficies Impermeables: Superficies construidas por el hombre a través de las cuales el agua no puede penetrar y que están por debajo de aproximadamente 2 metros de altura
9. Caminos Impermeables: Superficies impermeables utilizadas y mantenidas para el transporte.

10. Dosel de Árboles sobre Estructuras: Bosque o dosel de árboles que se superpone con superficies impermeables, haciendo que las estructuras no sean parcial o completamente visibles a simple vista.
11. Dosel de Árboles sobre Superficies Impermeables: Bosque o dosel de árboles que se superpone con superficies impermeables, lo que hace que la superficie impermeable no sea parcial o completamente visible a simple vista. Nota: las superficies impermeables y la copa de los árboles se mapearon de forma independiente; la copa de los árboles sobresalientes se identificó superponiendo estas clases para aislar las áreas de superposición.
12. Dosel de Árboles sobre Caminos Impermeables: Bosque o dosel de árboles que se superpone con superficies impermeables, haciendo que los caminos no sean parcial o completamente visibles a simple vista.
13. Aberdeen Proving Ground: Clase sin datos disponibles, solo existe en Harford/Maryland.

ÁREA DE ESTUDIO

El área de estudio es encuentra ubicada en la costa leste de los Estados Unidos (EE. UU.), siendo representada por 4 rectángulos circundantes, como se puede ver en el mapa de la Figura 5, lo que corresponde a un área de cerca de 46,66 km^2.

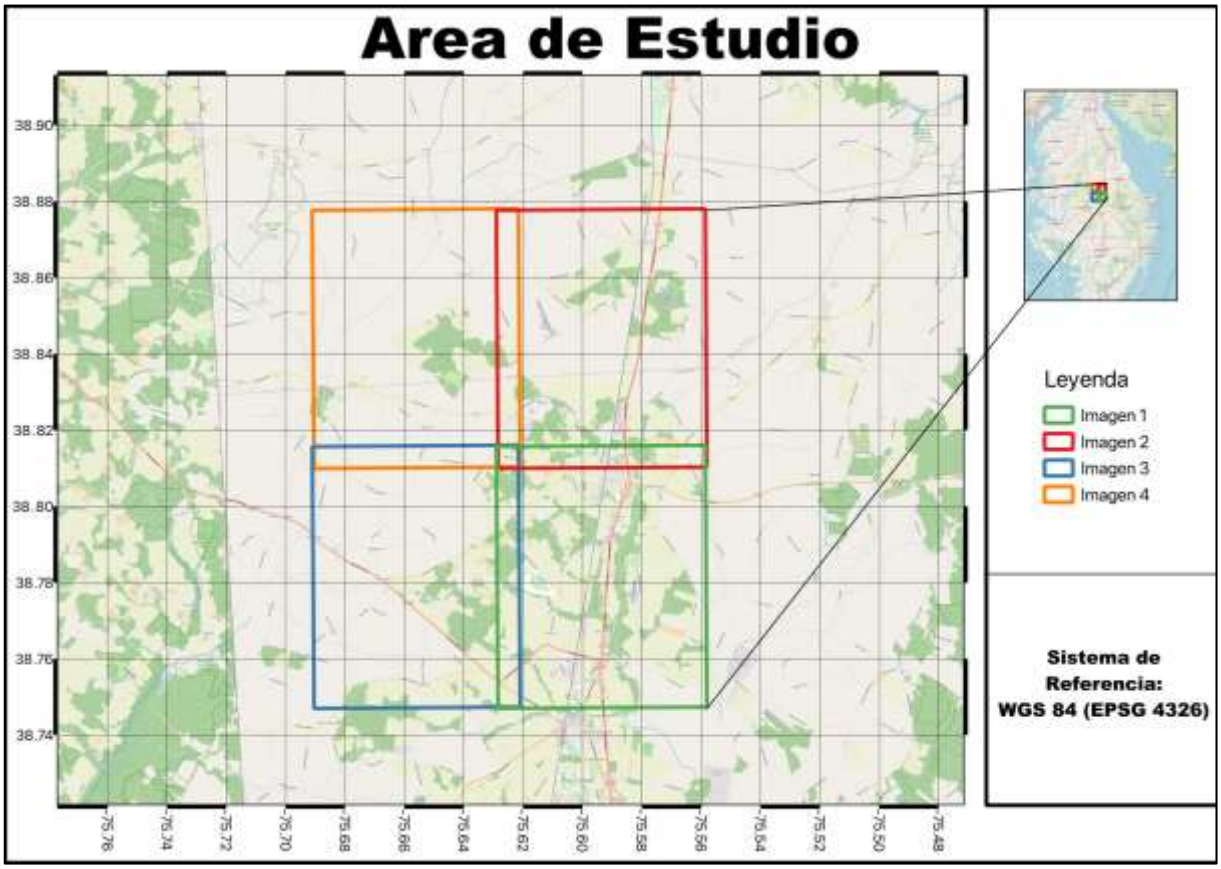

Figura 5 – Ubicación del Área de Estudio

CARGA DE DATOS

La carga de datos demanda el acceso a los archivos presentes en el conjunto de datos, es decir, las imágenes y etiquetas que serán utilizadas en la tarea de segmentación semántica. Para evaluar la calidad de los modelos entrenados, los datos se subdividieron en tres conjuntos de muestras: entrenamiento, validación y predicción. Se utilizaron muestreos aleatorios para el conjunto de entrenamiento, mientras que se utilizaron muestreos regulares para los conjuntos de validación y predicción.

La Figura 6 muestra la concepción general de la carga de datos para realizar la segmentación semántica con el conjunto de datos presentado en este Estudio de Caso.

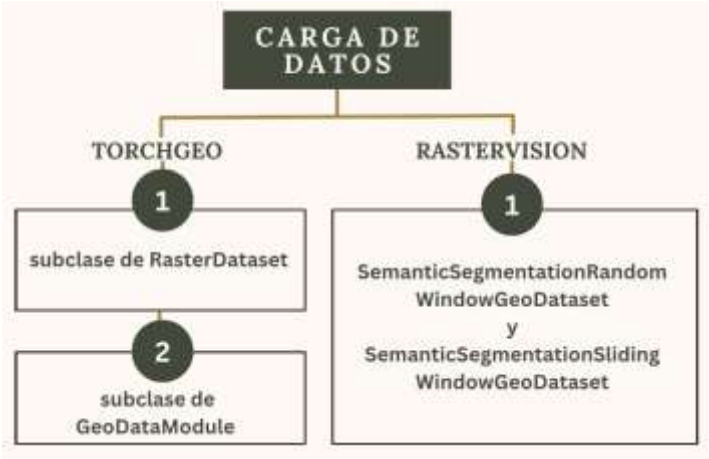

Figura 6 – Flujo de Carga de datos

Para TorchGeo, dado que el conjunto de datos utilizado es personalizado, se requiere la creación de subclases de *RasterDataset* para cargar las imágenes y las etiquetas utilizadas. Además, fue necesario crear una subclase de *GeoDataModule*, en la cual se define como particionar los datos para crear áreas que serán utilizadas en la tarea en cuestión. En nuestro enfoque de entrenamiento, se ha definido un área del 60% mediante muestreo aleatorio para el entrenamiento, y un área del 20% mediante muestreo regular para la validación y predicción.

En el caso de RasterVision, la creación de los datasets requiere que la forma de muestreo sea definida de antemano. En este sentido, los datos seleccionados para el entrenamiento (2 imágenes) se muestrean de manera aleatoria, mientras que los datos de validación (1 imagen) y predicción (1 imagen) se muestrean de manera regular. Para los datos de entrenamiento, se generan 2 *SemanticSegmentationRandomWindowGeoDataset*, que luego se concatenan mediante el uso de ConcatDataset, disponible en PyTorch. En cuanto a los datos de validación y predicción, se crean objetos *SemanticSegmentationSlidingWindowGeoDataset*.

CONFIGURACIÓN DEL ENTRENAMIENTO

La concepción general de la carga de datos para realizar la segmentación semántica con el conjunto de datos presentado en este Estudio de Caso puede ser vista en la

Figura 7.

Figura 7 – Flujo Configuración del Entrenamiento

Para configurar un entrenamiento en TorchGeo, es necesario definir una tarea. En nuestro estudio, hemos creado la tarea con el uso de *SemanticSegmentationTask*. Un *SemanticSegmentationTask* demanda la definición del modelo y *backbone* utilizados. Además, se debe especificar el número de canales (bandas espectrales), el número de clases, la función de perdida y la tasa de aprendizaje.

Cabe señalar que, actualmente, solo los modelos DeepLabV3+ (Chen *et al.*, 2018), Unet (Ronneberger, Fischer and Brox, 2015) y FCN (Shelhamer, Long and Darrell, 2016) pueden ser utilizados en tareas de segmentación semántica en TorchGeo. Los *backbones* suportados son los codificadores TIMM (Iakubovskii, 2023), lo que permite, por ejemplo, utilizar los *backbones* de la familia ResNet, como ResNet50 o ResNet101 (He *et al.*, 2016).

Para RasterVision, la configuración del entrenamiento se realiza utilizando un *SemanticSegmentationLearnerConfig*. Este, por su vez, demanda que otras configuraciones sean ingresadas. Las configuraciones de los datos son ingresadas con el uso de un *SemanticSegmentationGeoDataConfig*. Las configuraciones de tamaño de lote y tasa de aprendizaje son ingresadas por intermedio de un *SolverConfig.* Finalmente, las configuraciones de modelo son ingresadas por intermedio de un *SemanticSegmentationModelConfig*.

Es importante señalar que un *SemanticSegmentationModelConfig* requiere la definición del *backbone* que será utilizado. Actualmente, las tareas de segmentación semántica en RasterVision solo trabajan con el modelo DeepLabV3 (Chen *et al.*, 2017) y con los *backbones* ResNet50 y ResNet101.

ENTRENAMIENTO

La visión general de los elementos involucrados en el entrenamiento es presentada en la Figura 8.

Figura 8 – Flujo de Entrenamiento

TorchGeo permite el uso de Pytorch Lightning. Por lo tanto, considerando las ventajas obtenidas con su utilización (Miret *et al.*, 2022; Sawarkar, 2022), decidimos utilizar PyTorch Ligthning en nuestro estudio. En términos generales, Pytorch Lightning ofrece clases que facilitan los procedimientos de entrenamiento con PyTorch, lo que incluye formas transparentes de lidiar con aceleración gráfica y paralelismo.

El entrenamiento con Pytorch Lightning se hace instanciando un *Trainer*, que permite la definición de un *ModelCheckpoint,* que permite salvar el modelo utilizado y de un registrador de métricas. En el cuaderno implementado en este trabajo, se optó por utilizar un *CSVLogger*.

El *Trainer* de Pytorch Lightining permite manejar de manera transparente la aceleración gráfica. Para realizar el entrenamiento en tarjetas gráficas (GPU), basta definir ‘gpu’ como acelerador. Además, en el Trainer se especifica la cantidad máxima de épocas que será utilizada para realizar en entrenamiento.

El entrenamiento propiamente dicho es iniciado con el uso del método ‘*fit’* del *Trainer* instanciado. Las subclases de *GeoDataModule* y el *SemanticSegmentationTask* mencionadas anteriormente deben ser pasadas como parámetros para el método ‘fit’ para iniciar el entrenamiento.

Con RasterVision, el entrenamiento se lleva a cabo por intermedio de un *SemanticSegmentationLearner*. Los parámetros de ese objeto son los conjuntos de datos creados anteriormente (entrenamiento y validación) y la configuración de entrenamiento, que es creada con un *SemanticSegmentationLearnerConfig*.

El entrenamiento propiamente dicho se hace por intermedio del método ‘*train’*, que después de concluir, debe ter su estado de procesamiento guardado con ‘*save_model_bundle’*.

PREDICCIÓN GEORREFERENCIADA

Los elementos involucrados en la predicción georreferenciada pueden ser verificados en la Figura 9.

Figura 9 – Flujo de la Predicción georreferenciada

TorchGeo aún no tiene la capacidad nativa de realizar inferencias georreferenciadas. Por lo tanto, fueron creadas funciones para georreferenciar las predicciones con el uso de Rasterio (Gillies *et al.*, 2013).

La función '*create_in_memory_chip'* fue creada para solventar el georreferenciamiento, generando archivos GeoTiff en memoria para cada inferencia realizada. Sus parámetros incluyen una matriz que representa la inferencia realizada, una tupla con los valores para realizar la transformación de píxel para coordenadas de mundo, el sistema de referencia espacial utilizado (CRS), y si la construcción de las imágenes será codificada en RGB o en apenas una banda.

La función '*georreferenced_chip_generator'* fue creada específicamente para generar una lista de inferencias georreferenciadas y, por lo tanto, hace uso de '*create_in_memory_chip'*. Sus parámetros incluyen un *dataloader* con muestreo regular, el modelo utilizado, el sistema de referencia espacial utilizado (CRS), el tamaño de píxel, los colores, y un parámetro booleano que indica se esos colores deben ser utilizados para crear la imagen georreferenciada final.

Para fusionar todas las inferencias, fue implementada una tercera función (*merge_georreferenced_ships*) que utiliza la función '*merge'* de Rasterio.

Con estas funciones, es posible llevar a cabo el proceso completo de inferencia. Inicialmente, del *datamodule* creado anteriormente se obtiene el conjunto de datos de prueba. Con ese conjunto de datos se crea un *dataloader* de muestreo regular. Por cuestiones de capacidad de la tarjeta gráfica NVIDIA Tesla P100 utilizada (NVIDIA, 2023), el tamaño de cada muestra de inferencia fue limitado a 2048x2048 píxeles. Es en ese paso que los valores de cantidad de píxeles y CRS son utilizados.

RasterVision, por otro lado, tiene la capacidad nativa de realizar inferencias georreferenciadas. Para iniciar el proceso de inferencia, un *SemanticSegmentationLearner* debe ser cargado en modo de predicción a partir de los datos guardados al término de entrenamiento.

Para realizar las predicciones y guardarlas en disco, es necesario preparar las predicciones con el del método '*predict_dataset*' de *SemanticSegmentationLearner*. Su parámetro más importante es el conjunto de datos de predicción. El retorno de '*predict_dataset*' se ingresa en el método '*from_predictions'* de *SemanticSegmentationLabels*. El retorno de ese método son las etiquetas que pueden ser guardadas un archivo GeoTiff con el uso del método '*save',* que demanda, entre otros, un local en disco para guardar la imagen, y la matriz de transformación de píxeles para coordenadas de mundo (que es obtenida del *dataset* utilizado).

RESULTADOS Y DISCUSIONES

Las implementaciones de carga de datos con métodos nativos de ambas las bibliotecas generaron diferentes subdivisiones en los datos para crear las muestras de entrenamiento, validación y prueba. Para RasterVision, es necesario definir de antemano las imágenes/etiquetas que serán utilizados en las diferentes subdivisiones del conjunto de datos. Para TorchGeo, es posible cargar todas las imágenes/etiquetas y generar subdivisiones aleatorias en los datos, lo que ofrece una mayor flexibilidad.

En cuanto al tiempo de procesamiento, los resultados alcanzados son los presentes en la Tabla 1.

Tabla 1 – Rendimiento, en segundos, por fase, de cada biblioteca

Etapa	**TorchGeo (s)**	**RasterVision (s)**	**TG-RV (s)**
Entrenamiento	383,02	1696,80	-1313,78
Predicción	886,77	52,30	834,47
Georreferenciamiento	0,95	11,08	-10,13
Total	1270,74	1760,18	-489,44

El tiempo de entrenamiento de RasterVision supera significativamente al tiempo de entrenamiento de TorchGeo. Sin embargo, RasterVision demuestra una eficiencia mayor en las etapas de predicción y georreferenciamiento. En lo general, TorchGeo logró realizar todo en proceso de segmentación semántica 489,44 segundos más rápido que RasterVision, equivalente a aproximadamente 8 min considerando 15 épocas.

Para RasterVision, los valores de las pérdidas de entrenamiento y validación presentaron una caída similar y uniforme, como se puede ver en la Figura 10.

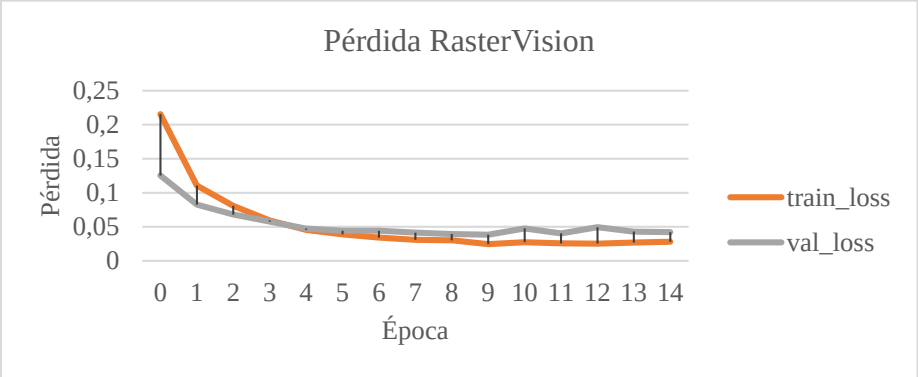

Figura 10 - Pérdida Entrenamiento/Validación de RasterVision

Para TorchGeo, los valores de las pérdidas de entrenamiento y validación presentaron una tendencia de caída, pero la pérdida del entrenamiento no tuvo una caída regular en todo en el procesamiento, como se puede ver en la Figura 11. Sin embargo, los valores calculados son mucho menores que los calculados por RasterVision.

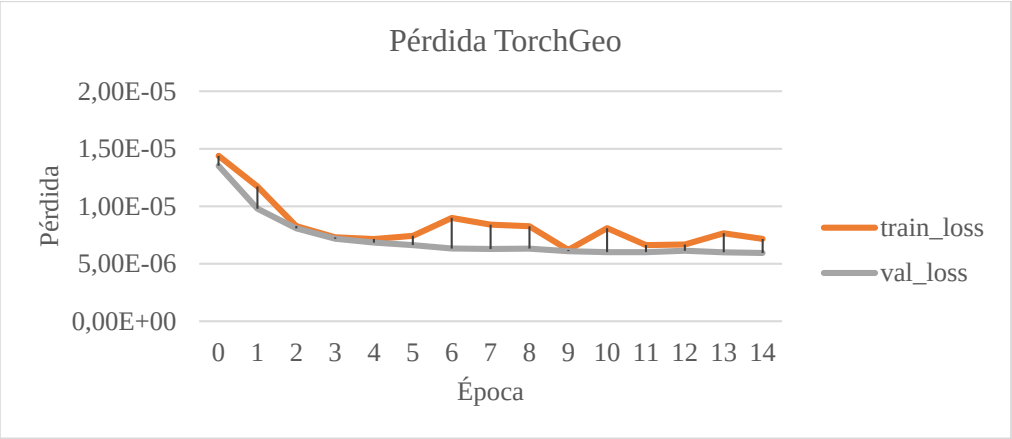

Figura 11 - Pérdida Entrenamiento/Validación de TorchGeo

Las métricas que RasterVision utiliza nativamente son F1, Precision y Recall (Goutte and Gaussier, 2005). Para el caso del presente estudio, los valores obtenidos son los presentes en la Figura 12.

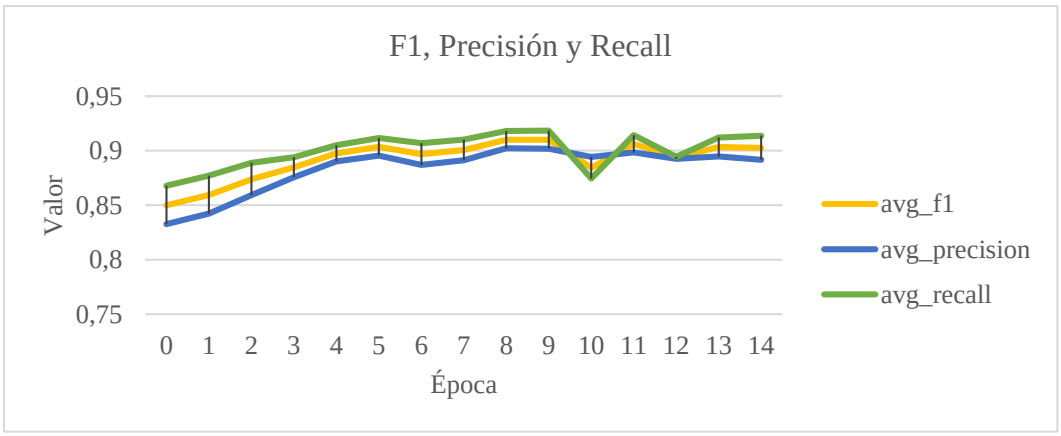

Figura 12 – Métricas Nativas de RasterVision

Para TorchGeo, las métricas nativas ofrecidas son exactitud (Heras, 2020) multiclase y índice de Jaccard (Costa, 2021) multiclase, como se puede ver en la Figura 13.

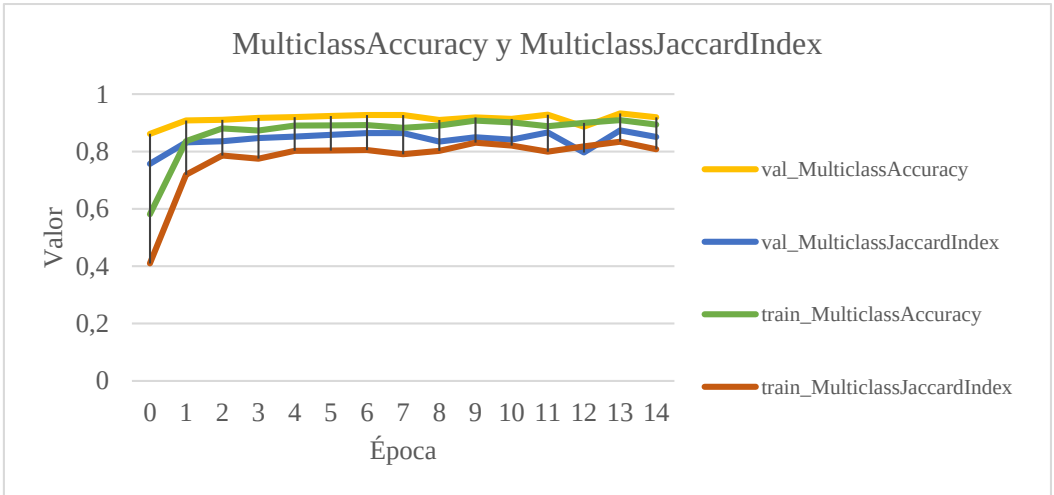

Figura 13 – Métricas Nativas TorchGeo

Para realizar una comparación más uniforme, se intersecaron las etiquetas y las predicciones georreferenciadas de ambas bibliotecas. Esto permitió realizar un análisis cualitativo y cuantitativo considerando la misma área.

Desde una perspectiva cualitativa, las predicciones de RasterVision pueden considerarse más adecuadas que las de TorchGeo, especialmente considerando el procesamiento en 15 épocas. En la Figura 14, se presentan las etiquetas comparadas con ambas predicciones, y la leyenda (14.d) está incluida para facilitar la identificación de las clases segmentadas.

La predicción de TorchGeo (14.b) en general muestra segmentaciones con bordes más ásperos. En áreas urbanizadas, presenta dificultades para identificar clases como “Estructuras” y “Caminos Impermeables”. Por otro lado, la predicción de RasterVision (14.c) exhibe segmentaciones con bordes más suaves. Algunas clases, como “Dosel de Árboles”, se expandieron, mientras que otras, como “Agua”, se comprimieron en comparación con las etiquetas ground truth (14.a).

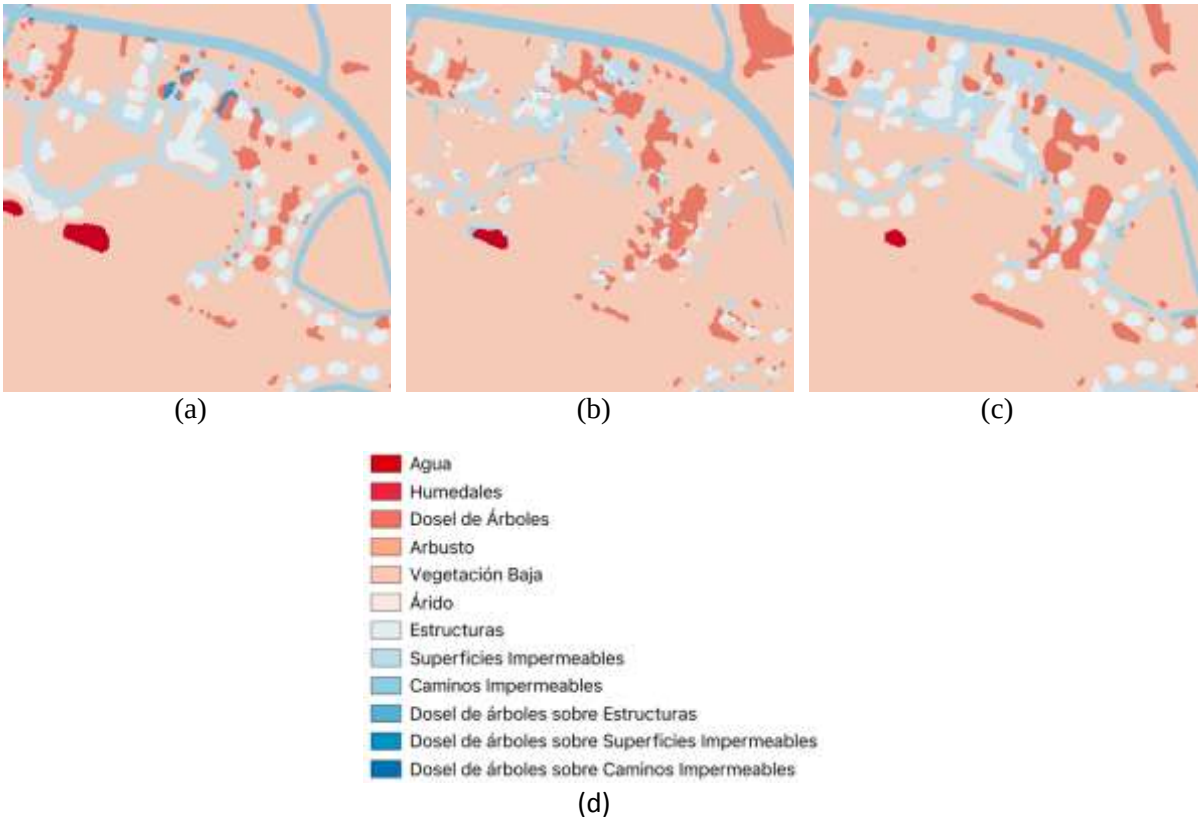

Figura 14 - Comparación entre predicciones

En términos cuantitativos, los valores de las métricas exactitud, precisión, recall, F1 score y Jaccard score (Tabla 2) confirman que las predicciones de RasterVision arrojaron mejores resultados que las predicciones de TorchGeo.

Tabla 2 – Valores de Exactitud, Precisión, Recall, F1 score y Jaccard

Biblioteca	Exactitud	Precisión	Recall	F1 score	Jaccard score
TorchGeo	91,82%	90,78%	91,82%	90,90%	85,06%
RasterVision	92,95%	92,84%	92,95%	92,84%	87,92%

Para explorar de manera más detallada los resultados de TorchGeo y RasterVision y para poder analizar cuantitativamente la Figura 14, se calculó la matriz de confusión entre ambas predicciones y las etiquetas, como se muestra en la Figura 15. Del análisis de ambas matrices, se desprende que los modelos entrenados presentaron comportamientos similares.

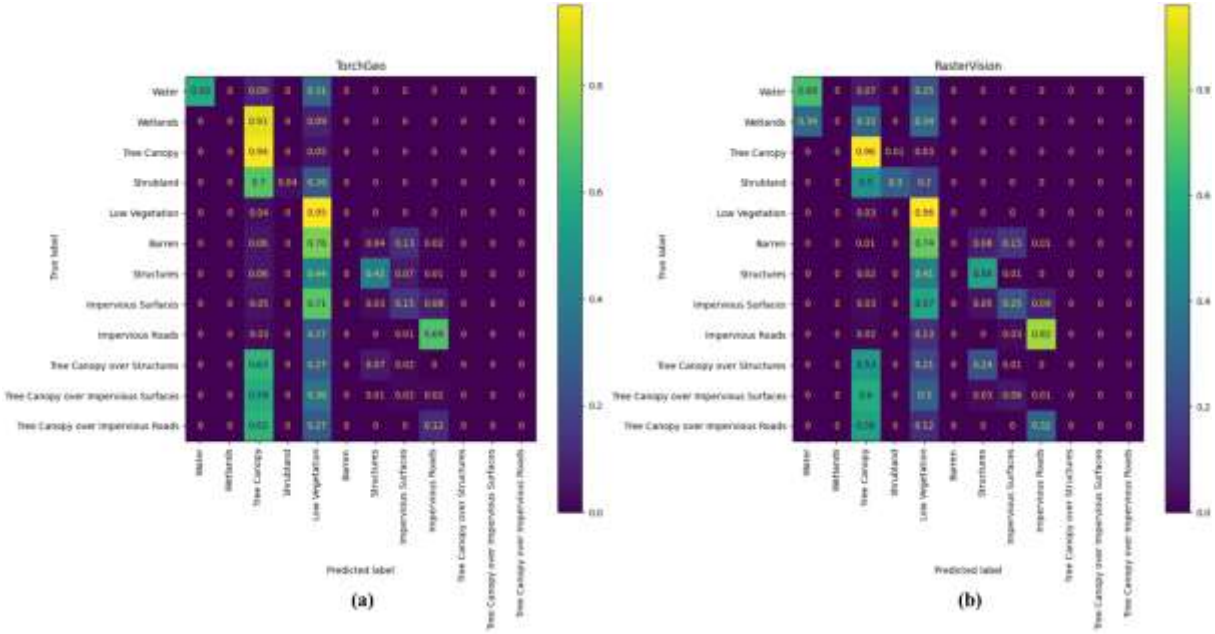

Figura 15 - Matrices de confusión

Tanto TorchGeo cuanto RasterVision experimentaron dificultades de distinguir la clase "Vegetación Baja" (*Low Vegetation*) de las demás. De manera similar, ambos modelos tuvieron problemas para distinguir la clase "Dosel de árboles" (*Tree Canopy*) de otras clases que presentan vegetación, como las clases "Dosel de Árboles sobre Estructuras" (*Tree Canopy over Structures*), "Dosel de Árboles sobre Superficies Impermeables" (*Tree Canopy over Impervious Surfaces*), "Dosel de Árboles sobre Caminos Impermeables" (*Tree Canopy over Impervious Roads*), "Humedales" (*Wetlands*) y "Arbusto" (*Shrubland*).

Es importante destacar que el modelo de RasterVision presentó resultados menos precisos al distinguir clases que incluyen "Caminos Impermeables" y "Estructuras", mientras que TorchGeo mostró dificultades para distinguir la clase "Dosel de árboles" (*Tree Canopy*) de las clases "Humedales" (*Wetlands*) y "Arbusto" (*Shrubland*).

En cuanto a los valores verdaderamente positivos, es decir, la diagonal principal, RasterVision exhibió valores superiores a los obtenidos por TorchGeo, lo que se refleja en la Tabla 2.

Es relevante señalar que las diferencias presentadas pueden estar asociadas a las diferencias de modelo (DeepLabV3 y DeepLabV3+) y con las diferentes subdivisiones generadas en el conjunto de datos.

TorchGeo utiliza DeepLabV3+, mientras que RasterVision utiliza DeepLabV3. DeepLabV3+ extiende DeepLabV3 al incorporar un módulo decodificador que refina resultados de segmentación, especialmente a lo largo de los límites de los objetos (Chen *et al.*, 2018), lo que puede ser una fuente de diferencias en las métricas calculadas nativamente.

Por otro lado, RasterVision trabajó con una subdivisión de datos basada en los archivos presentes en el conjunto de datos (2 imágenes para entrenamiento, 1 imagen para validación y 1 imagen para predicción), mientras que TorchGeo trabajó con una subdivisión aleatoria del

conjunto de datos (60% para entrenamiento, 20% para validación y 20% para predicción). Esta variación en la configuración espacial también puede ser una fuente de las diferencias observadas en las métricas calculadas.

Es relevante señalar que, de manera distinta, otras investigaciones que procesan datos de sensores remotos, como satélites y drones, como los trabajos de Zhang *et al.* (2020), Zhao *et al.* (2022), Dong, Pan and Li (2019), Ding, Tang and Bruzzone (2021), Kampffmeyer, Salberg and Jenssen (2016) y Kemker, Salvaggio and Kanan (2018), se centran en analizar y proponer nuevos modelos o mejoras de modelos existentes para llevar a cabo el procesamiento de datos geoespaciales.

Estas implementaciones están adaptadas a sus respectivas investigaciones, lo que les permite utilizar modelos que podrían arrojar resultados más efectivos. En otras palabras, sus implementaciones emplean métodos y modelos que están el estado del arte. No obstante, estas implementaciones no son accesibles para investigadores que no sean expertos en Aprendizaje Profundo. Además, dado que el enfoque de estas investigaciones se centra en los modelos utilizados, no existe una preocupación por la generación de predicciones georreferenciadas. El interés principal radica en obtener métricas de evaluación cada vez mejores.

CONCLUSIONES

En este trabajo, se realizaron dos procesos distintos de segmentación semántica de imágenes de satélite utilizando dos bibliotecas diferentes: TorchGeo y RasterVision. Al emplear un conjunto de datos común, se llevó a cabo una comparación detallada de la complejidad de implementación y el tiempo de ejecución de cada tarea.

En términos de documentación, ambas las bibliotecas ofrecen API una extensa documentación y tutoriales que apoyan los primeros pasos de profesionales que desean utilizarlas. Además, los equipos de desarrollo de ambas bibliotecas han demostrado ser ágiles en responder preguntas y comprometidos en interactuar con los usuarios.

El análisis de este estudio revela que ambas bibliotecas cumplen con los objetivos propuestos, demostrando su capacidad para manejar las particularidades de los datos geoespaciales. Pueden manejar de manera efectiva imágenes multiespectrales y diferentes sistemas de referencia espacial, siendo opciones viables para profesionales de geoprocesamiento que buscan aplicar técnicas de aprendizaje profundo en datos geoespaciales sin perder sus propiedades críticas.

No obstante, es importante destacar que RasterVision si distingue por su funcionamiento más transparente, ya que puede realizar inferencias georreferenciadas de manera nativa. Por otro lado, RasterVision se mostró menos flexible que TorchGeo en la carga de datos, requiriendo, en la versión utilizada, un preprocesamiento adicional en conjunto de datos utilizado.

TorchGeo, en cambio, ofrece una gran flexibilidad para cargar datos, incluso permitiendo trabajar con imágenes y etiquetas que no comparten los mismos sistemas de referencia espaciales, límites espaciales, tamaño de píxeles y bandas espectrales. Además, la capacidad de generar conjunto de datos de unión o intersección mediante el uso de los operadores “|” y “&” hacen con que la carga de datos de TorchGeo sea más versátil.

Por otro lado, es importante señalar que TorchGeo aún no tiene la capacidad nativa de realizar una inferencia georreferenciada. Por lo tanto, es necesario tener un trabajo de implementación extra para que las inferencias puedan ser georreferenciadas. Siendo así, a depender de las características del conjunto de datos utilizado, el georreferenciamiento de las inferencias podría volverse complejo.

En cuanto a la eficiencia de procesamiento, se observa que TorchGeo llevó 1270,74 segundos, mientras que RasterVision tomó 1760,18 segundos para realizar el entrenamiento,

predicción y georreferenciamiento de las inferencias. Esto representa un aumento de un 38% en el tiempo de procesamiento, se convirtiendo en una diferencia de cerca de 8 min, lo que puede ser considerada significativo para el estudio de caso presentado.

A pesar del mayor tiempo de procesamiento, RasterVision presentó mejores valores para todas las métricas calculadas (exactitud, precisión, recall, F1 score y Jaccard score), lo que puede ser considerado un factor decisivo para los profesionales del ámbito geoespacial.

En conclusión, considerando las versiones actuales utilizadas (TorchGeo 0.4.1 y RasterVision 0.21), se puede concluir que la utilización de RasterVision se vuelve más recomendada para expertos de geociencias que no sean necesariamente expertos en programación. A pesar de que RasterVision presenta un tiempo de procesamiento 38% superior, su capacidad nativa de realizar inferencias georreferenciadas y los resultados superiores en las métricas utilizadas se destacan como aspectos clave para su elección. Por otro lado, TorchGeo aún no dispone de la capacidad nativa de realizar inferencias georreferenciadas. Aunque requiera la implementación de funciones de georreferenciamiento adicionales, entregó un tiempo de procesamiento superior.

Sin embargo, es importante señalar que este panorama puede cambiar en el futuro. El equipo de desarrollo de TorchGeo está trabajando en la implementación de mecanismos transparentes para que los usuarios puedan realizar inferencias georreferenciadas de manera más eficiente, lo que podría ofrecer tiempos de procesamiento más rápidos.

Finalmente, ambas herramientas cumplen con el desafío de procesar datos geoespaciales y pueden considerarse un puente entre el campo de la geoinformación y el campo de la Inteligencia Artificial. La elección de cual utilizar debe basarse en la experiencia de cada investigador, considerando los aspectos presentados en este estudio.

REFERENCIAS

Andrade, L. C. O. de (2023a) *RasterVision 101*. Available at: https://www.kaggle.com/code/luizclaudioandrade/rastervision-101/ (Accessed: 29 August 2023).

Andrade, L. C. O. de (2023b) *TorchGeo 101*. Available at: https://www.kaggle.com/code/luizclaudioandrade/torchgeo-101/ (Accessed: 29 August 2023).

Azavea (2022) *RasterVision*. Available at: https://github.com/azavea/raster-vision (Accessed: 26 August 2023).

Bojer, C. S. and Meldgaard, J. P. (2020) 'Kaggle forecasting competitions: An overlooked learning opportunity'. doi: 10.1016/j.ijforecast.2020.07.007.

Borges, L. E. (2014) *Python para desenvolvedores: aborda Python 3.3*. Novatec Editora.

Borovec, J. *et al.* (2022) *Pytorch Lightning*. doi: 10.5281/zenodo.7447212.

Chen, L.-C. *et al.* (2017) 'Rethinking Atrous Convolution for Semantic Image Segmentation'. Available at: http://arxiv.org/abs/1706.05587.

Chen, L. C. *et al.* (2018) 'Encoder-decoder with atrous separable convolution for semantic image segmentation', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11211 LNCS, pp. 833–851. doi: 10.1007/978-3-030-01234-2_49.

Chesapeake Bay Program Office (CBPO) (2022) *One-meter Resolution Land Cover Dataset for the Chesapeake Bay Watershed, 2013/14*. Available at: https://www.chesapeakeconservancy.org/conservation-innovation-center/high-resolution-data/lulc-data-project-2022/ (Accessed: 27 August 2023).

Congedo, L. (2021) 'Semi-Automatic Classification Plugin: A Python tool for the download and processing of remote sensing images in QGIS', *Journal of Open Source Software*, 6(64), p. 3172. doi: 10.21105/joss.03172.

Costa, L. da F. (2021) 'Further Generalizations of the Jaccard Index'. Available at: http://arxiv.org/abs/2110.09619.

Dillon, J. V. *et al.* (2017) 'TensorFlow Distributions'. Available at: http://arxiv.org/abs/1711.10604.

Ding, L., Tang, H. and Bruzzone, L. (2021) 'LANet: Local Attention Embedding to Improve the Semantic Segmentation of Remote Sensing Images', *IEEE Transactions on Geoscience and Remote Sensing*, 59(1), pp. 426–435. doi: 10.1109/TGRS.2020.2994150.

Dong, R., Pan, X. and Li, F. (2019) 'DenseU-Net-Based Semantic Segmentation of Small Objects in Urban Remote Sensing Images', *IEEE Access*, 7, pp. 65347–65356. doi: 10.1109/ACCESS.2019.2917952.

Gillies, S. *et al.* (2013) *Rasterio: Geospatial raster I/O for Python programmers*. Available at: https://github.com/rasterio/rasterio (Accessed: 5 September 2023).

Goutte, C. and Gaussier, E. (2005) 'A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation', *Lecture Notes in Computer Science*, 3408(January), pp. 345–359. doi: 10.1007/978-3-540-31865-1_25.

He, K. *et al.* (2016) 'Deep residual learning for image recognition', *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem, pp. 770–778. doi: 10.1109/CVPR.2016.90.

Heras, J. M. (2020) *Precision, Recall, F1, Accuracy en clasificación*. Available at: https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/ (Accessed: 6 September 2023).

Iakubovskii, P. (2023) *TIMM Enconders*. Available at: https://smp.readthedocs.io/en/latest/encoders_timm.html (Accessed: 30 August 2023).

Kampffmeyer, M., Salberg, A. B. and Jenssen, R. (2016) 'Semantic Segmentation of Small Objects and Modeling of Uncertainty in Urban Remote Sensing Images Using Deep Convolutional Neural Networks', *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 680–688. doi: 10.1109/CVPRW.2016.90.

Kemker, R., Salvaggio, C. and Kanan, C. (2018) 'Algorithms for semantic segmentation of multispectral remote sensing imagery using deep learning', *ISPRS Journal of Photogrammetry and Remote Sensing*. Elsevier, 145, pp. 60–77. doi: 10.1016/J.ISPRSJPRS.2018.04.014.

Ketkar, N. *et al.* (2021) 'Introduction to pytorch', *Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch*. Springer, pp. 27–91.

Konecny, G. (2014) *Geoinformation: remote sensing, photogrammetry and geographic information systems*. cRc Press.

Lecun, Y., Bengio, Y. and Hinton, G. (2015) 'Deep learning', *Nature*, 521(7553), pp. 436–444. doi: 10.1038/nature14539.

Miret, S. *et al.* (2022) 'The Open MatSci ML Toolkit: A Flexible Framework for Machine Learning in Materials Science'. Available at: http://arxiv.org/abs/2210.17484.

Nachmany, Y. and Alemohammad, H. (2019) 'Detecting roads from satellite imagery in the developing world', *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2019-June, pp. 83–89.

National Aeronautics and Space Administration (2020) *National Agriculture Imagery Program (NAIP)*.

NVIDIA (2023) *NVIDIA Tesla P100*. Available at: https://www.nvidia.com/en-us/data-center/tesla-p100/ (Accessed: 5 September 2023).

O'Mahony, N. *et al.* (2020) 'Deep Learning vs. Traditional Computer Vision', *Advances in Intelligent Systems and Computing*, 943(Cv), pp. 128–144. doi: 10.1007/978-3-030-17795-9_10.

Quarati, A., De Martino, M. and Rosim, S. (2021) 'Geospatial open data usage and metadata quality', *ISPRS International Journal of Geo-Information*. MDPI AG, 10(1). doi: 10.3390/ijgi10010030.

Ronneberger, O., Fischer, P. and Brox, T. (2015) 'U-Net: Convolutional Networks for Biomedical Image Segmentation', *arXiv*, pp. 1–8. Available at: http://lmb.informatik.uni-freiburg.de/%0Aarxiv:1505.04597v1.

Sawarkar, K. (2022) *Deep Learning with PyTorch Lightning: Swiftly Build High-performance Artificial Intelligence (AI) Models Using Python*. Packt Publishing Ltd.

Shelhamer, E., Long, J. and Darrell, T. (2016) 'Fully Convolutional Networks for Semantic Segmentation'.

Stevens, E., Antiga, L. and Viehmann, T. (2020) 'Pretrained networks', in *Deep learning with PyTorch*. Manning Publications, pp. 46–68.

Stewart, A. J. *et al.* (2021) 'TorchGeo: Deep Learning With Geospatial Data'.

Stewart, A. J. *et al.* (2022) 'TorchGeo: deep learning with geospatial data', in *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*. Association for Computing Machinery. doi: 10.1145/3557915.3560953.

Zhang, J. *et al.* (2020) 'Multi-scale context aggregation for semantic segmentation of remote sensing images', *Remote Sensing*, 12(4), pp. 1–16. doi: 10.3390/rs12040701.

Zhao, Q. *et al.* (2022) 'Semantic Segmentation With Attention Mechanism for Remote Sensing Images', *IEEE Transactions on Geoscience and Remote Sensing*, 60, pp. 1–13. doi: 10.1109/TGRS.2021.3085889.