Trabajo de Fin de Grado

Clasificación semántica de nubes de puntos LiDAR mediante el uso de redes neuronales

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Jorge Armas Morales

TUTORIZADO POR: Adrián Peñate Sánchez, Agustín Trujillo Pino y Sebastián Eleazar Ortega Trujillo

Fecha 01/2025

DIRECTORA DE LA ESCUELA DE INGENIERÍA INFORMÁTICA

Agradecimientos

A Chano, por ayudarme a escapar de las coordenadas soviéticas.

A Adrián, Agustín y Cristina por guiar el barco a puerto

A Marci y Esteban, por sentarme en su mesa.

A Toño y a Pablo, por bajarse al barro

A Carlos, vida antes que muerte.

Y a mi padre

Resumen

En este trabajo se han entrenado diferentes modelos diferentes de la arquitectura de la Pointet++. El objetivo de estos modelos es el de segmentar semánticamente diferentes nubes de puntos LiDAR. Los modelos han sido construidos siguiendo una metodología incremental de sprints e involucrando cuatro conjuntos de datos diferentes. Tras realizar una serie de experimentos se han obtenido dos modelos competentes que aprovechan la potencia del transfer learning y los datasets DALES y ECLAIR.

Abstract

In this proyect we have trained a set of Pointnet++ models. The main objective behind these models is to segment different LiDAR point clouds. For the construction of the models it has been used an agile metodology based of sprints involving four different datasets. After a series of experiments we have obtained two different and competent models that make use of transfer learning and DALES's and ECLAIR's datasets

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Objetivo

Este proyecto surge como extensión de la oferta de prácticas externas publicada por Aerolaser System S.L. La meta principal de este proyecto es entrenar un modelo de red neuronal capaz de identificar correctamente los diferentes elementos de una escena aérea de tendido eléctrico tomada por AeroLaser, especialmente la distinción entre vegetación e infraestructura eléctrica.

1.2. Competencias específicas

A continuación describimos las competencias trabajadas durante el desarrollo de este TFT:

- ✔ 1. TFG: "Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas."
- ✔ 2. CI15: "Conocimiento y aplicación de los principios fundamentales y técnicas básicas de los sistemas inteligentes y su aplicación práctica"
- ✔ 3. FB1: "Capacidad para la resolución de los problemas matemáticos que puedan plantearse en la ingeniería. Aptitud para aplicar los conocimientos sobre: álgebra lineal, cálculo diferencial e integral, métodos numéricos, algorítmica numérica, estadística y optimización."
- ✔ 2. FB4: "Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación de la ingeniería."

- ✔ 3. FB5: "Conocimiento de la estructura, organización, funcionamiento e interconexión de los sistemas informáticos, los fundamentos de la programación, y su aplicación para la resolución de problemas propios de la ingeniería."
- ✔ 4. CI6: "Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos."
- ✔ 5. CI7: "Conocimiento, diseño y utilización de forma eficiente de los tipos y estructuras de datos..

1.3. Principales aportaciones

Este proyecto surge con la idea de facilitar el análisis de la infraestructura de tendido eléctrico en España, mejorando y abaratando por consiguiente el mantenimiento y transporte de la energía eléctrica. Por esto se establece una relación alta con el noveno objetivo de desarrollo sostenible y una relación casi tangencial con el séptimo.

Citando la página de las Naciones Unidas[18] en el artículo relativo al objetivo de desarrollo 9 Ïnvertir en infraestructuras —transporte, regadío, energía y tecnologías de la información y la comunicación— es crucial para lograr un desarrollo sostenible y empoderar a las comunidades de muchos países.".

Con esto llegamos a la conclusión de que todo aquello que favorezca el correcto desarrollo y mantenimiento de las infraestructuras nacionales (públicas o privadas) supone un aporte de cara a la sostenibilidad a lato plazo. Esto se ve reflejado en el cuadro 1.1.

1.4. Metodología

Este proyecto ha utilizado una metodología incremental partiendo de un análisis de los objetivos del proyecto, los recursos disponibles y el estado del arte de las tecnología involucradas tales como arquitecturas de red neuronal convolutivas, librerías para el manejo de datos LiDAR(laspy [4]) y librerías para la optimización del uso de datos(NumPy [6]).

Una vez concluido esta primera etapa comenzamos con un periodo formativo con el objetivo de adquirir los conocimientos necesarios para poder arrancar con el trabajo, esto incluye la familiarización con el trato de datos LAS, el aprendizaje de técnicas de preprocesado de datos y la familiarización con PyTorch en general y la red escogida en particular.

Entrando al fin en el desarrollo de los diferentes modelos este ha seguido una estructura incremental por "sprints". El objetivo de esta metodología es generar un modelo que pueda ser entrenado con diferentes combinaciones de datasets para obtener el modelo entrenado que mejor responda ante los datos de AeroLaser [2].

Cuadro 1.1: Grado de relación del TFT con los objetivos de desarrollo sostenible.

ODS	Grado de relación			
	0 No procede	1 Bajo	2 Medio	3 Alto
1 Fin de la Pobreza	X			
2 Hambre cero	X			
3 Salud y Bienestar	X			
4 Educación de calidad	X			
5 Igualdad de género	X			
6 Agua limpia y saneamiento	X			
7 Energía Asequible y no contaminante		X		
8 Trabajo decente y crecimiento económico	X			
9 Industria, Innovación e Infraestructuras				X
10 Reducción de las desigualdades	X			
11 Ciudades y comunidades sostenibles	X			
12 Producción y consumo sostenibles	X			
13 Acción por el clima	X			
14 Vida submarina	X			
15 Vida de ecosistemas terrestres	X			
16 Paz, justicia e instituciones sólidas	X			
17 Alianzas para lograr objetivos	X			

1.5. Recursos hardware

Para realizar este proyecto se ha contado con 3 equipos de diferente potencia. El primero de ellos es el ordenador proporcionado por AeroLaser durante la estancia de prácticas.

A la conclusión de las prácticas fueron usados 2 equipos, un ordenador particular y un equipo de laboratorio proporcionado por la Escuela de Ingeniería Informática. Las prestaciones de estos equipos se resumen en la tabla a continuación:

Cuadro 1.2: Características de los equipos usados

	PC doméstico	PC del laboratorio
GPU	NVIDIA GeForce 1060 (3GB VRAM)	NVIDIA GeForce RTX 4090 (56GB VRAM)
RAM	16GB	64GB
CPU	AMD Ryzen 7 1700 Eight-Core 8 GHz	13th Gen Intel(R) Core(TM) i9-13900K 3.00 GHz

De cara a cualquiera que desee entrenar en la versión final de este programa con los datos de AeroLaser(los más pesados) sepa que son necesarios 30 GB de VRAM y 24 de RAM

1.6. Recursos software

En el apartado del software ha sido necesaria una variada colección de herramientas

Si las dividimos por tecnologías, tendríamos lo siguiente:

- ✔ PyCharm: IDE y debug
- ✔ Anaconda: Software de gestión de entornos virtuales para la instalación de dependencias
- ✔ Pytorch: Framework en el que se han entrenado y evaluado todos los modelos
- ✔ NumPy: Creación y uso de estructuras de datos optimizadas en memoria
- ✔ LasPy: Tratamiento de ficheros en formato LAS y LAZ
- ✔ TQDM: Mejora visual de la interfaz
- ✔ Open3D: Reconstrucción visual de escenas tridimensionales
- ✔ MatLab: Uso puntual para la comprobación del funcionamiento de la Pointnet++. Licencia proporcionada por la Universidad de Las Palmas de Gran Canaria

Capítulo 2

Estado del arte

2.1. Introducción

El objetivo de este capítulo es ahondar en las principales tecnologías utilizadas en este proyecto y sus características más relevantes.

En este capítulo se resume una de las principales tecnologías en topografía, llamada LiDAR, el problema troncal de este proyecto en la segmentación semántica, las principales arquitecturas de red para resolverlo y se establecerá una comparativa entre estas últimas.

2.2. LiDAR

El LiDAR (Light Detection and Ranging o Laser Imaging Detection and Ranging[12]) es una tecnología estándar en el campo de la topografía. Esta mide la distancia a un objeto mediante la medición de la demora de un haz de láser, ya sea pulsado o continuo(también llamado de medición de fase).

Ampliando el tamaño del espacio de muestras se puede obtener un conjunto de datos que representen de manera fidedigna un espacio tridimensional (figura 2.1).

Existen a su vez cuatro tipos de LiDAR aéreos diferentes en función del trazado que dibuja el haz láser en la toma de mediciones. Conste aquí que solo se explican los tipos de LiDAR aéreos dado que así han sido tomados todos los datos usados en este proyecto (figura 3.9).

Estos cuatro tipos serían el lineal, zig-zag, elíptico o Palmer y de fibra óptica (figura 2.2). En el caso de AeroLaser se ha utilizado el de tipo lineal [9] montado sobre un helicóptero que ha recogido datos de cientos de escenas aéreas de tendido eléctrico de alta tensión en territorio nacional.

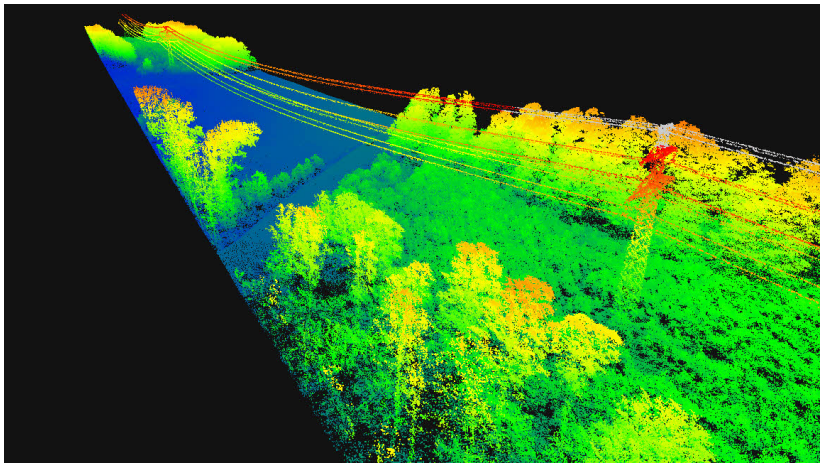

Ilustración 2.1: Escena de tendido eléctrico tomada con LiDAR

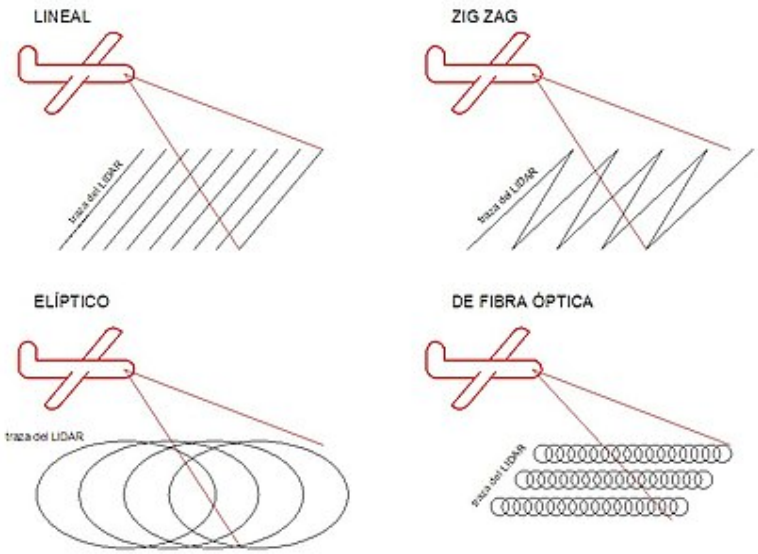

Ilustración 2.2: Esquema de los cuatro grandes tipos de LiDAR Aéreo

Estos son los datos que nos interesa segmentar y para los que buscaremos la mejor combinación entre arquitectura de red y conjunto de datos para el preentrenamiento.

Ilustración 2.3: LiDAR en helicóptero tomando escenas de poblado. Cortesía de AeroLaser System S.L.

2.3. Segmentación Semántica

La segmentación semántica es uno de los grandes problemas arquetípicos de la inteligencia artificial en general y del aprendizaje profundo en particular.

Ésta consiste en catalogar todos los puntos de una imagen o escena tridimensional en una de varias clases con el objetivo de poder localizar todos los elementos diferentes en la región de interés.

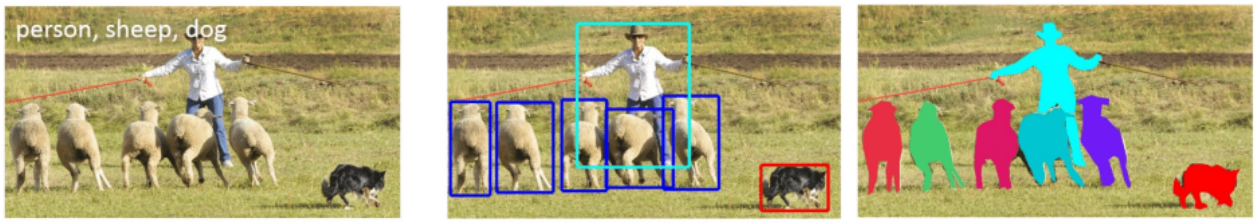

Ilustración 2.4: Comparativa de funcionamiento entre clasificación y segmentación. De izquierda a derecha, imagen en bruto, clasificación y segmentación

Podría afirmarse que esto no es otra cosa que el problema de clasificación tradicional aplicado a cada punto (hablando de entornos tridimensionales). Esto no dista de la verdad pero tratarlo en esto términos no deja de ser una sobre-simplificación (comparativa en la figura 2.4).

La segmentación entraña problemas propios no presentes en la clasificación. Uno de estos es la gran variedad de tamaños de entrada que puede presentar una nube de puntos tomada a partir de una escena real.

Para esto existe un procedimiento llamado downsampling. Éste es el proceso de reducción de la cantidad de datos de un conjunto a una cantidad concreta, mediante diversas técnicas. Esto entraña el riesgo de perder información relevante por el camino.

Para paliar esto se suele tomar la mayor cantidad de datos posibles, buscando provocar la suficiente redundancia para volver a los datos robustos frente a reducciones de resolución y selecciones aleatorias.

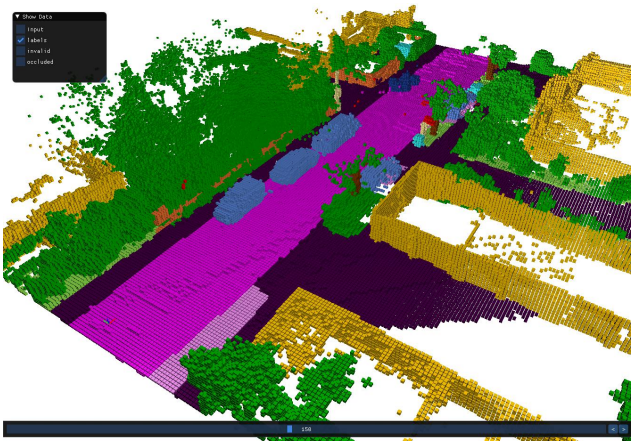

Ilustración 2.5: Ejemplo de segmentación semántica en escena tridimensional de exterior.

Este problema aparece con frecuencia, entre muchos otros, en los siguientes campos:

- ✔ Conducción autónoma: Detección de elementos de la calzada, además de peatones y vehículos (ilustración 2.5).
- ✔ Medicina: Identificación de tejidos dañados o detección temprana de tumores.
- ✔ Inspección industrial: Análisis en busca de defectos mecánicos.
- ✔ Topografía: Análisis de la evolución de la vegetación en un área o asistencia en la delimitación de parcelas.

Debido a la naturaleza costosa de las redes neuronales todo escalado que se realice ha de ser consciente y con un razonamiento detrás. Es por esto que a continuación que exploran arquitecturas con las que enfrentar estos problemas.

Podemos encontrar ejemplos de segmentación semántica, sus principales problemas y el desempeño de diferentes arquitecturas para enfrentarlos en los papers «A Brief Survey on Semantic Segmentation with Deep Learning»[20] y «A review of semantic segmentation using deep neural networks»[15]. El primero de estos fue publicado en el año 2020 por Shijie Hao, Yuan Zhou, y Yanrong Buo en el número 406 de la revista «Neurocomputing». El segundo de los papers fue publicado por Yanming Guo, Yu Liu, Theodoros Georgiou y Michael S. Lew como parte del «International Journal of Multimedia Information Retrieval» en 2018

2.4. PointNet

El 10 de abril de 2017 fue publicada en la Conference on Computer Vision and Pattern Recognition (CVPR) la arquitectura de red de la Pointnet [28]. Ésta es una arquitectura de red neuronal que pretende servir como una herramienta de propósito general en la clasificación semántica.

Esto se debe a que, a diferencia de otras populares arquitecturas, ésta no requiere del proceso de voxelizado de la nube de puntos (agrupación de puntos cercanos en unidades volumétricas o vóxeles que sintetiza la principal información del conjunto), sino que admite los datos desordenados. Esta característica la hace ideal para un entorno tan variable como son las escenas aéreas de tendido eléctrico.

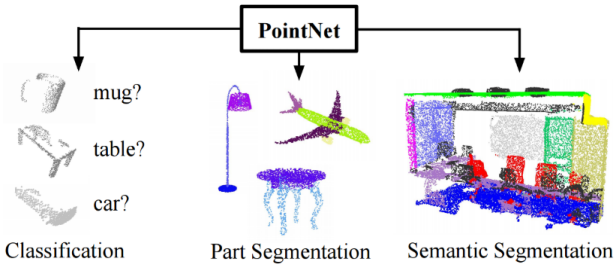

Ilustración 2.6: Resumen de las tres vertientes de la Pointnet. La segmentación por partes es un caso específico de la segmentación semántica

La manera de lograr la asimilación de puntos de manera caótica radica en la arquitectura de la red, citando el paper: “The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. “mlp” stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.” [28].

Entrando en detalle la Pointnet[28], consta de dos tipos de redes, clasificación y segmentación(ilustraciones 2.6 y 2.7), aunque sería más correcto afirmar que la segunda utiliza de la primera para su funcionamiento. El interés para el proyecto radica en la segunda, pero será necesario explicar la clasificación para entender la segmentación.

La Pointnet[28] acepta como entrada cualquier cantidad de puntos tridimensionales que transforma en un espacio uniforme mediante el uso de lo que el paper llama una T-Net. Esto no es otra cosa que un sector que transforma la entrada en una matriz cuadrada de orden 3. Esta matriz a su vez es multiplicada por la matriz de entrada (figura 2.8).

El resultado de este proceso es suministrado a un segmento de identificación de características en el que se generan un número de vectores de 64 elementos igual a la cantidad de puntos de entrada N (figura 2.9).

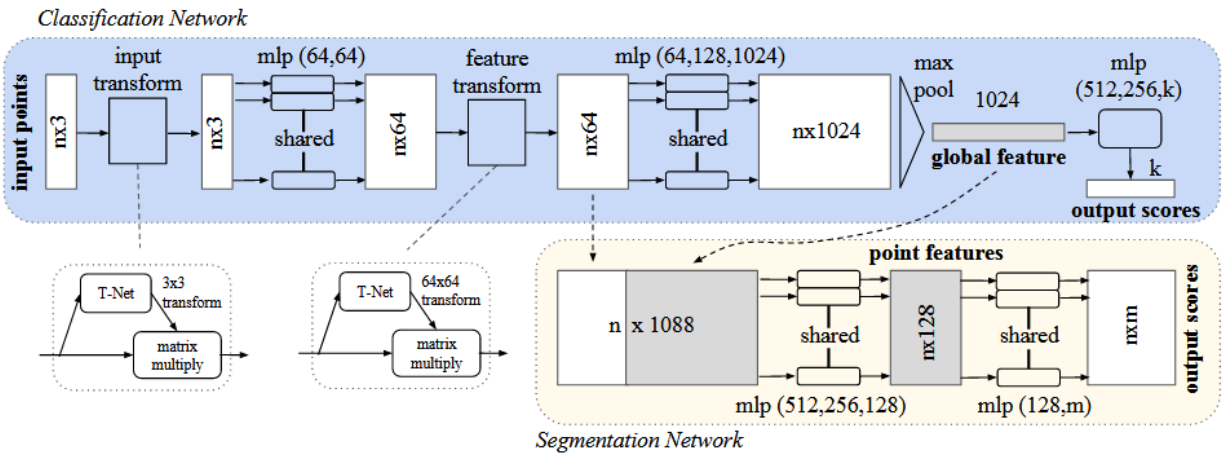

Ilustración 2.7: Arquitectura general de la Pointnet. Imagen cortesía de la universidad de Stanford

La salida de esta etapa esencial es una matriz de dimensiones Nx64 a la que se aplica el mismo proceso que en el párrafo anterior, pero generando una matriz de orden 64 en este caso.

Esta nueva matriz resulta de especial interés puesto que será utilizada en breve cuando se explique la segmentación. Mientras tanto, el siguiente paso consiste en enfrentar dicha matriz a un perceptrón que mediante convolución amplía sus dimensiones de 64 a 128 a 1024.

A continuación se hace pasar a la matriz de Nx1024 por un proceso de max-pooling , es decir una red neuronal completamente conectada con funciones de activación no lineal[28], que se encarga de extraer las características más pronunciadas.

En este punto se produce la diferencia fundamental entre la red de clasificación y la de segmentación. En la primera, después del proceso de max-pooling se utiliza un último perceptrón que devuelve un vector en el que solo se identifica una clase. En cambio en la arquitectura de segmentación se produce una ampliación o concatenación de la matriz Nx64 y la resultante del proceso de max-pooling, formando un resultado de Nx1088.

Basándose en el razonamiento de que la segmentación no es más que el problema clásico de clasificación aplicado a cada punto, se usa esta matriz nueva con el objetivo de combinar la información individual de cada punto con la información global de la red.

Por último, esta matriz de información combinada se le hace pasar por un proceso inverso de perceptrones completamente conectados que van reduciendo su tamaño hasta dar lugar a una matriz que asigna una única clase a cada uno de los puntos por lo que sus dimensiones corresponden en un eje a N y en el otro al número de clases (figura 2.10).

Pese a su aparente idealidad, la PointNet es bastante sensible al desbalanceo de clases. Esto fue comprobado por A. Nurunnabi , F. N. Teferle , J. Li , R. C. Lindenbergh y S. Parvaz en el artículo “Investigation of pointnet for semantic segmentation of large-scale outdoor point clouds”[26], citando textualmente “Likewise, the 1st experiment, the results reveal that the accuracy of the point classification is subjugated by the classes having more training points”.

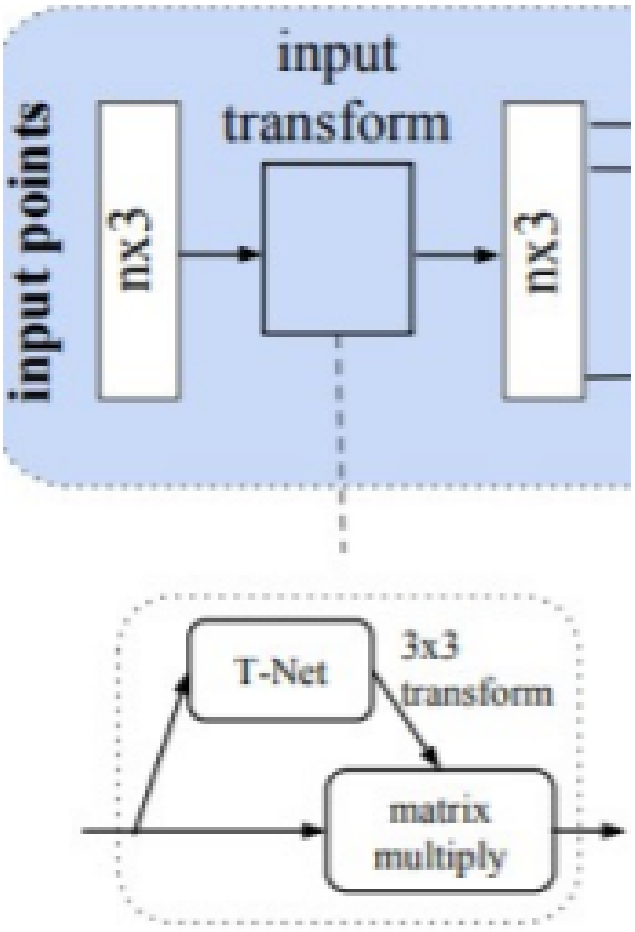

Ilustración 2.8: Entrada y transformación de la entrada

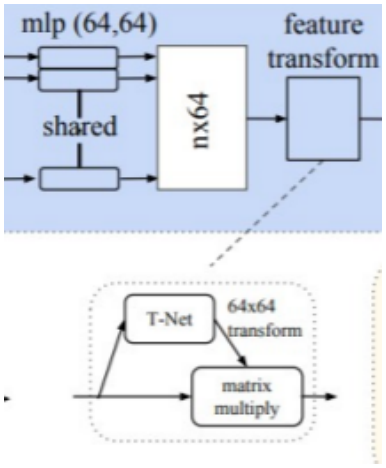

Ilustración 2.9: Perceptrón multicapa (MLP) y modificación de la salida

Esta debilidad presente en la PointNet nos lleva a pensar en utilizar otra arquitectura para este problema pese a su idealidad en datos irregulares.

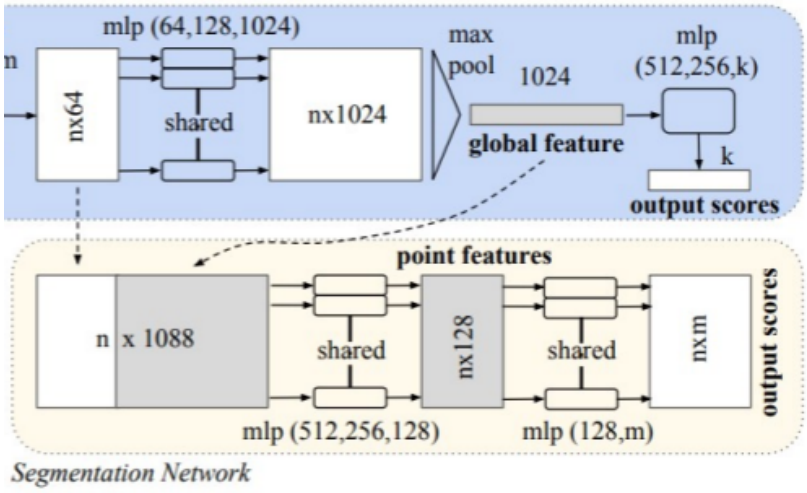

Ilustración 2.10: Ampliación de la red de clasificación para la segmentación

2.5. KPConv

Dentro de las arquitecturas de red para la clasificación y segmentación de nubes de puntos, la antes nombrada PointNet supuso una innovación de gran carácter y sentó la base para otras que han seguido. Una de ellas, y que según estudios comparativos [31] supera a su inspiración, es KPConv[32] (abreviatura de Kernel Point Convolution). Esta fue publicada en la International Conference of Computer Vision (ICCV) de 2019.

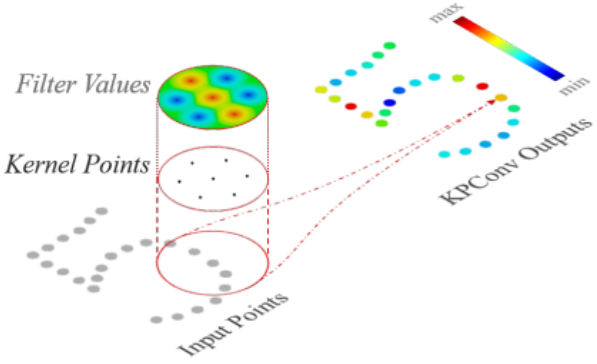

Ilustración 2.11: Abstracción del funcionamiento de KPConv. Cortesía de Mines Paris Tech, Facebook AI Research y Stanford University

Esta arquitectura se basa en la extracción de elementos mediante el uso de puntos clave o “kernel points”[32]. Estos son todos aquellos puntos en los que se centra un radio concreto llamado kernel. Puede observarse una abstracción de este funcionamiento

El objetivo detrás del uso de zonas de radio definido en vez de la tradicional ”K Nearest Neighbours.º KNN (K vecinos más cercanos) es el de aportar robustez frente a densidades de puntos altamente variables, siendo éste uno de los principales defectos de las nubes de

puntos.

El uso de estos kernels está pensado para cubrir con ligero solape todo el espacio de datos. Esto entraña ventajas ya comentadas y problemas, tales como la dificultad de establecer una correlación entre puntos de kernel y sus vecinos si estos no se encuentran en zona de solape.

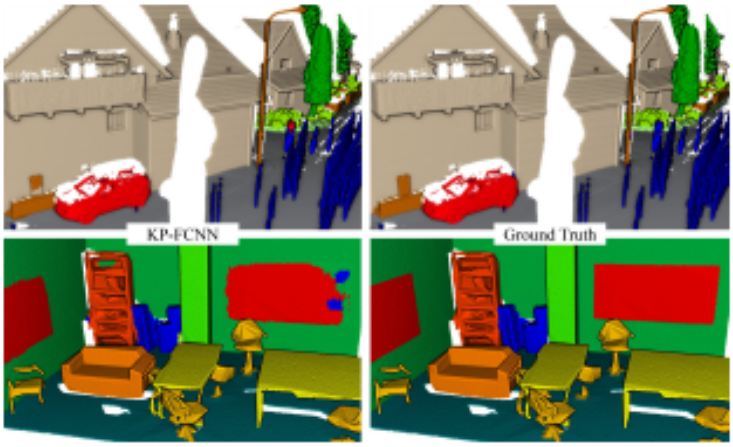

Figure 4. Outdoor and Indoor scenes, respectively from Semantic3D and S3DIS, classified by KP-FCNN with deformable kernels.

Ilustración 2.12: Resultado de la clasificación de Semantic3D(arriba) y S3DIS(abajo) con KPConv con kernels deformables

Para esto se establece una relación lineal de cada punto con sus vecinos, de manera que se pueda identificar un punto de manera contextual. Cuanto más cerca se encuentren dos puntos cualesquiera, mayor será su relación lineal. Esto refuerza a KPConv[32] en el reconocimiento de estructuras y patrones relevantes.

Una vez se ha establecido esa correlación lineal, ésta se multiplica por la matriz de pesos. Esto permite a la red combinar mayor información para una mejor extracción de características. El funcionamiento de esto resulta extremadamente similar al sistema de filtros de una red convolutiva para reconocimiento de imágenes tradicional (figura 2.13).

A su vez, KPConv[32] incluye dos tipos diferentes de kernels. Hasta ahora, la explicación venía centrada en los kernels rígidos pero existe la posibilidad de utilizar los llamados deformables. Estos son una ampliación de los kernels rígidos consistente en aplicar una fuerza repulsiva desde el punto de kernel (que se mantiene inmóvil) y otra atractiva para mantenerlos dentro del radio del kernel.

Esto ha demostrado aumentar gratamente la eficacia de KPConv[32] (figura 2.12), pero supone un gran aumento en el costo computacional, quedando reservado el uso de los kernels deformables a los proyectos de gran potencia y con experiencia, puesto que la regulación de los parámetros de deformación requiere un control exhaustivo.

El proceso al completo para cada punto de kernel consiste en generar su radio (o esfera) y localizar los puntos dentro de dicho kernel. En siguiente etapa se establece la correlación lineal

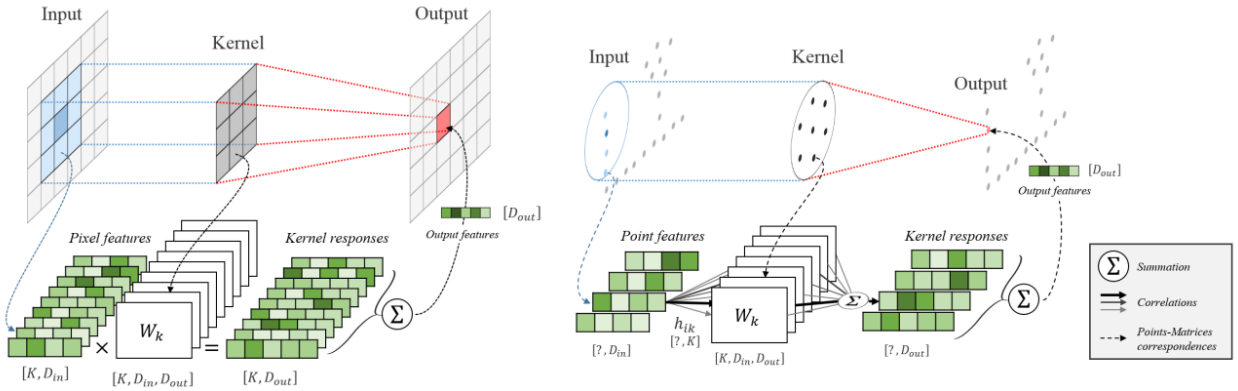

Ilustración 2.13: Comparativa entre clasificación de imágenes tradicional y KPConv. Cortesía de Mines Paris Tech, Facebook AI Research y Stanford University

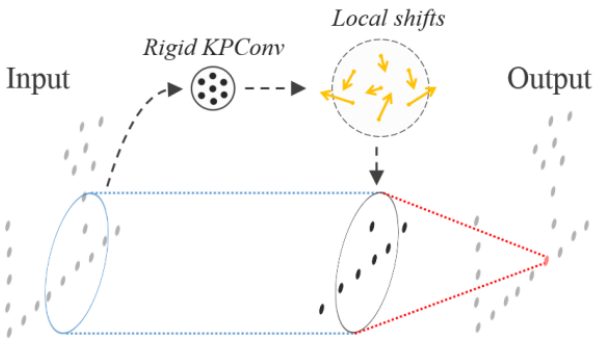

Ilustración 2.14: Representación gráfica de los Kernels deformables. Cortesía de Mines Paris Tech, Facebook AI Research y Stanford University

entre las características del punto kernel y sus vecinos y eso es multiplicado por la matriz de pesos general. Tras obtener las características de todos los puntos dentro del kernel, esto se suma y con ello obtenemos la información final de cada punto.

2.6. PointNet++

La PointNet++ es una arquitectura de red neuronal convolutiva que utiliza una estructura jerárquica que le permite reconocer características de las escenas tridimensionales con mayor detalle que su predecesor [28]. Esta arquitectura es una ampliación directa que muestra una mejora notable en los resultados incluyendo la segmentación semántica [29] y fue publicada como parte del Advances in Neural Information Processing Systems 30 en 2017.

La arquitectura de la Pointnet++ es una ampliación directa de su predecesora, basada en el uso de tres capas diferente. Estas serían, en orden: una de muestreo, otra de agrupamiento y otra de Pointnet[28].

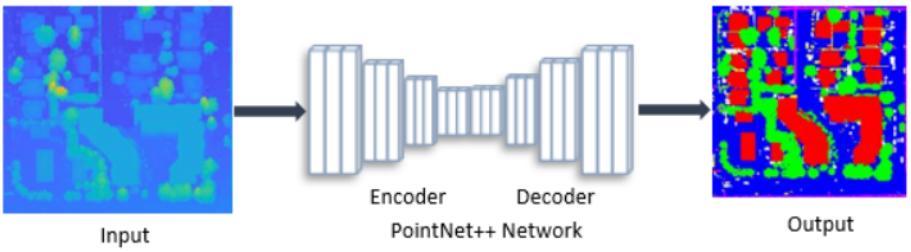

Ilustración 2.15: Abstracción del funcionamiento de la Pointnet++. Cortesía de MatLab

Las tres capas son utilizadas, en el orden antes explicado, varias veces con el objetivo de extraer una síntesis de la información del conjunto de entrada (bidimensional o tridimensional). Esta síntesis será utilizada para clasificación o segmentación en función del proceso que sigamos a continuación.

La capa de muestreo o sampling se encarga de generar un conjunto de datos más acorde a la distribución de los datos que el muestreo aleatorio. Si tomamos los puntos de entrada, podemos utilizar Farthest Point Sampling para generar un conjunto de puntos que garantizan estar a la máxima distancia posible entre ellos. A estos puntos los llamaremos centroides.

La capa de agrupación se encarga de generar sectores dentro del conjunto total de puntos en base a los centroides antes obtenidos y una distancia de radio fija. Debido a que la cantidad de puntos en cada grupo varía la capacidad de la Pointnet[28] de adaptarse a valores de entrada diferentes, resulta ideal.

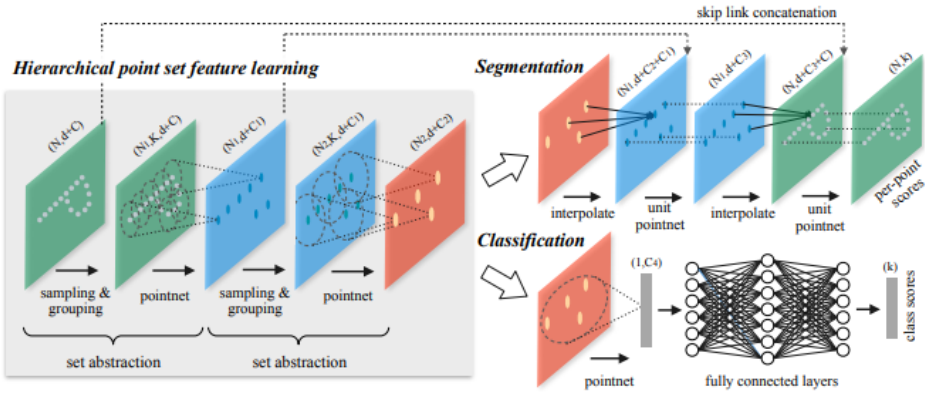

Ilustración 2.16: Detalle de la estructura de la Pointnet++. Cortesía de Stanford University

La tercera capa es una Pointnet[28] que actuará sobre cada sector antes generado de manera que se obtengan una serie de regiones preclasificadas o segmentadas que representen la principal información de la nube de puntos. Este proceso de tres capas se ejecuta dos veces de manera consecutiva (figura 2.16) y aquí es donde se produce la bifurcación en función del problema que se quiera resolver.

En el caso de la clasificación, el vector extraído de la última Pointnet[28] se hace pasar por un conjunto de capas fully connected para extraer un vector one hot (un valor o clase a

1, el resto a 0) que nos dice la clase predicha en cuestión.

Por el lado de la segmentación, en cambio, ese vector pasa por un proceso de interpolación y el resultado se concatena con los resultados de la segunda capa en orden inverso (observar las flechas en la figura 2.18). El resultado de esto se utiliza como entrada a capa de Pointnet[28] sin la parte final de max pooling explicado en la sección 2.4. Este proceso de interpolación, concatenación y Pointnet[28] se repite otra vez devolviendo una matriz con la clasificación final de cada uno de los puntos.

La diferencia entre la estructura jerárquica del sucesor, combinada con la estrategia clásica de divide y vencerás, y el modelo global de la arquitectura original, es lo que provoca que la Pointnet++ sea una de las opciones más viables para el análisis semántico de nube de puntos tridimensionales de datos LiDAR.

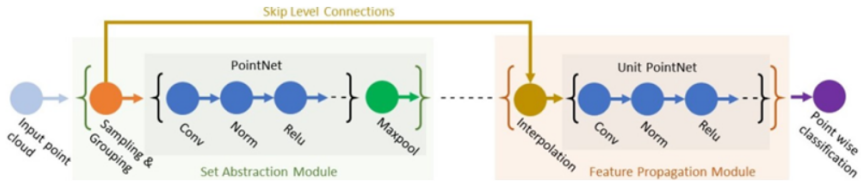

Ilustración 2.17: Estructura general de la segmentación semántica con Pointnet++. Cortesía de MatLab

2.7. Comparativa

Tras analizar en detalle tres diferentes arquitecturas de red, quedaría pendiente establecer, de manera empírica, cuál resulta en un mejor desempeño frente al tipo de datos que maneja este proyecto.

Este análisis comparativo ya fue realizado por la Universidade de Vigo en el año 2020. En este estudio, llamado "SEMANTIC SEGMENTATION OF POINT CLOUDS WITH POINTNET AND KPCONV ARCHITECTURES APPLIED TO RAILWAY TUNNELS"[31], compararon las métricas resultantes de enfrentar a la Pointnet[28] y a KPConv[32] con un conjunto de datos propio.

Haciendo uso de un conjunto de datos representando escenas tridimensionales de túneles ferroviarios, consiguieron comprobar que KPConv[32] respondía mejor que Pointnet[28] en todos los casos (figura 2.20).

La Pointnet[28] fue entrenada y validada con el conjunto de datos, pero con KPConv[32] se siguieron pasos extra. Primero se realizó un proceso de transfer learning desde el dataset de Semantic3D[19] y de manera paralela se generó un modelo de KPConv[32] entrenado dos veces sobre el conjunto de datos de la Universidade de Vigo (KPConv TL y KPConv R en la figura 2.20 respectivamente).

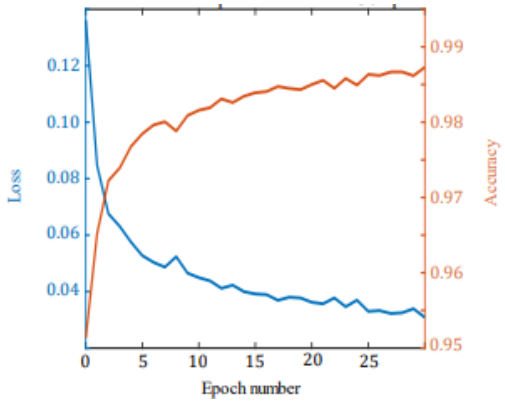

Ilustración 2.18: Relación entre Loss y Accuracy en el entrenamiento de la Pointnet. Cortesía de la Universidade de Vigo

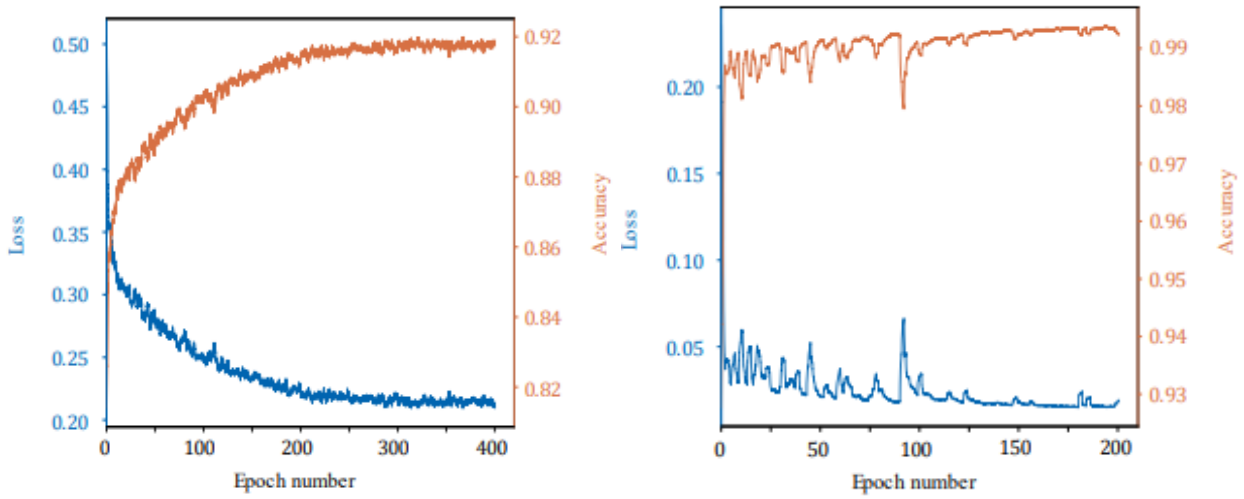

Ilustración 2.19: Relación entre Loss y Accuracy en el entrenamiento de KPConv TL (izq) y KPConv R (der)

Al comparar los tres modelos diferentes, vemos cómo KPConv[32] se alza por encima de la Pointnet[28] en cuanto al Valor-F1 o F1-score (media armónica entre la precisión y la exhaustividad) y mantiene unos resultados similares entre las dos formas de entrenamiento. Cabe destacar que el tiempo de entrenamiento entre todos ellos resulta hasta tres veces más rápido, por época, para la Pointnet[28] debido al proceso de subsampling ya explicado.

	Precision			Recall			F1-score		
	KPConv TL	KPConv R	PointNet	KPConv TL	KPConv R	PointNet	KPConv TL	KPConv R	PointNet
Ground	99.22%	98.84%	99.15%	96.69%	94.75%	91.22%	**97.94%**	96.75%	95.02%
Lining	98.55%	99.87%	98.02%	99.81%	99.51%	99.66%	99.18%	**99.69%**	98.83%
Wiring	98.83%	99.99%	100%	87.42%	92.15%	87.66%	92.77%	**95.91%**	93.42%
Rails	87.82%	39.38%	42.18%	89.35%	100.00%	99.90%	**88.58%**	56.51%	59.33%

Ilustración 2.20: Comparativa entre KPConv y Pointnet. Cortesía de la Universidade de Vigo

Citando el trabajo realizado por Martin Kada y Dmitry Kuramin en "ALS Point Cloud Classification using PointNet++ and KPConv[32] with Prior Knowledge"[21], KPConv[32] se plantea como una arquitectura más que capaz de lidiar con el problema presente que se ve apenas superada por la Pointnet++.

La diferencia principal radica en la IoU (intersección sobre unión), la cual resulta la métrica más relevante para este problema.

Pese a que KPConv[32] lleva la delantera en algunas de las clases, las más relevantes como terreno y vegetación caen frente a la Pointnet++.

Es observable que la mayor ventaja de KPConv[32] radica en su capacidad de identificar de manera algo más fiable las clases menos representadas. Aún con eso, no resulta suficiente para considerarla mejor alternativa que el sucesor de la Pointnet[28] para este proyecto.

	Prior			IoU							
Method	HaG	Cls	OA	*mean*	*ground*	*vegetation*	*roof*	*facade*	*vehicle*	*line*	*pole*	*error*
PointNet++			0.931	0.605	0.919	0.836	0.903	0.703	0.291	0.707	0.170	0.310
	×	×	0.964	0.691	**0.981**	0.919	0.932	0.721	0.432	0.852	0.184	0.506
PointNet++ (+2D)	×		0.935	0.615	0.925	0.837	0.918	0.716	0.135	**0.877**	0.189	0.322
	×	×	**0.968**	**0.702**	0.980	**0.921**	**0.946**	**0.782**	**0.434**	0.814	**0.224**	**0.516**
KPConv			**0.955**	0.624	0.942	0.893	0.945	0.793	0.502	0.577	0.069	0.268
	×		0.949	0.684	0.932	0.891	0.921	0.792	0.498	**0.898**	0.223	0.319
		×	0.954	0.669	**0.948**	0.882	**0.948**	0.773	0.377	0.878	**0.238**	0.309
	×	×	0.953	**0.686**	0.938	**0.898**	0.930	**0.779**	**0.506**	0.878	**0.238**	**0.321**

Ilustración 2.21: Métricas enfrentadas entre KPConv y Pointnet++. Cortesía de Technische Universität Berlin, Instituto de Geodesia y Ciencias de la Geoinformación, Berlín, Alemania

Capítulo 3

Conjuntos de datos

3.1. Descripción de los datasets

Para la realización de este proyecto han sido utilizados cuatro datasets diferentes. Estos son y en orden de uso: Un conjunto de datos proporcionado por Aerolaser System S.L., S3DIS, DALES y ECLAIR.

El primero de los datasets es aquel que más nos interesa debido al contexto en el que se desarrolla este proyecto, a su vez es el data set más desbalanceado en cuanto a sus clases. Presenta seis catalogadas como suelo, vegetación, edificaciones, torres eléctricas, cableado y fondo.

El data set de Stanford 3D fue utilizado únicamente con objetivo de probar que la arquitectura de PointNet++ funcionaba correctamente en Pytorch. Esta dataset es proporcionado por la Universidad de Stanford bajo la licencia del MIT y contiene nubes de puntos tridimensionales que describen una escena de interior. Este dataset, aunque completo y equilibrado, no contiene datos de interés para el modelo a entrenar; por este mismo motivo, su uso ha sido anecdótico.

Por otro lado, el data set ECLAIR presenta un equilibrio de clase mayor que el de AeroLaser y datos de interés al ser escenas de tendido eléctrico. El mayor atractivo de este dataset radica en la robustez y variedad en los datos debido al uso de pseudoetiquetas, esto es, clasificaciones extraídas de un modelo preentrenado y no anotadas a mano.

Por último se ha utilizado el dataset DALES, que es aquel con el que el proyecto original de la Pointnet++ en MatLab entrena y evalúa la red. Este data set presenta un equilibrio de clases mayor y unos datos relevantes para el problema a resolver.

3.2. S3DIS

S3DIS es un conjunto de siglas correspondiente a Stanford 3D Indoor Scene. Este dataset contiene escenas tridimensionales de interior; en total son 6 grandes áreas con 271 habitaciones. En estas habitaciones, cada punto está anotado como perteneciente a una de las 13 clases diferenciadas.

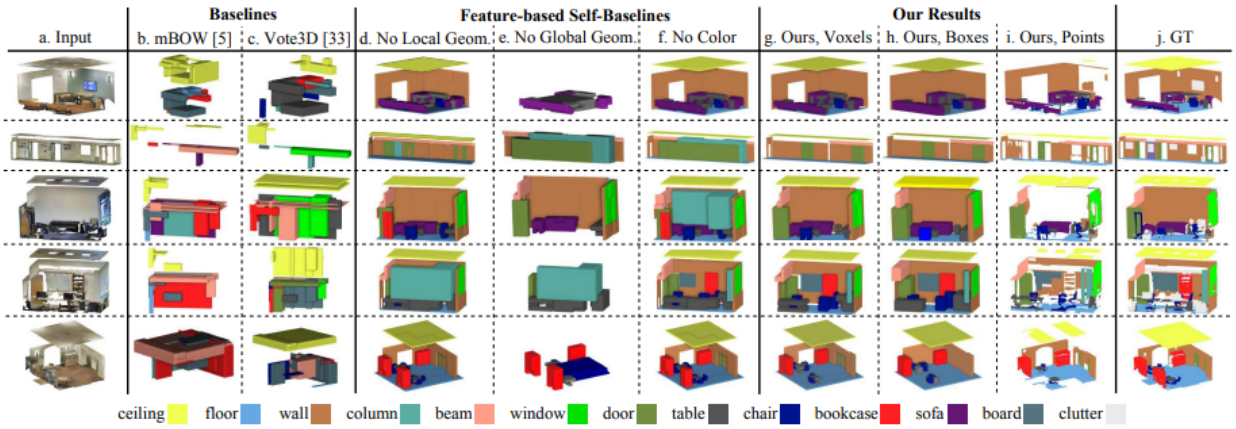

Ilustración 3.1: Resumen de las escenas de interior del dataset S3DIS y sus clases. Cortesía de Stanford University

Este dataset surge bajo el amparo de la universidad de Stanford y es uno de los referentes en cuanto a escenas de interior. Por esto es el utilizado como marco de referencia para las pruebas de segmentación semántica de la PointNet++.[29].

Las clases corresponden a una variedad de elementos propios de interior, tales como el techo, suelo, muros, columnas, vigas, ventana, puertas, mesas, sillas, estanterías, sofás, pizarras y la clase desorden que sirve para todo aquello que no encaje en las demás. Esto puede verse representado en en la figura 3.1

Si observamos los datos presentes en la figura 3.2, observamos cómo existe un desbalanceo claro hacia las clases correspondientes a los elementos estructurales (techo, suelo, muros, columnas, vigas, ventanas y puertas) sobre aquellas relacionadas con el mobiliario (mesas, sillas, estanterías, sillones, pizarra y desorden).

Aunque este dataset es uno de los mejores en su campo actualmente, este proyecto necesita de entrenamiento en escenas de exterior, con escenas rurales a ser posible y unas clases similares a las del dataset de AeroLaser y por este motivo su uso ha sido puntual.

Cabe decir que en el momento de redactar esta memoria el repositorio original de S3DIS y las páginas asociadas se encuentran caídas. En caso de que vuelvan a funcionar el último enlace funcional es este.

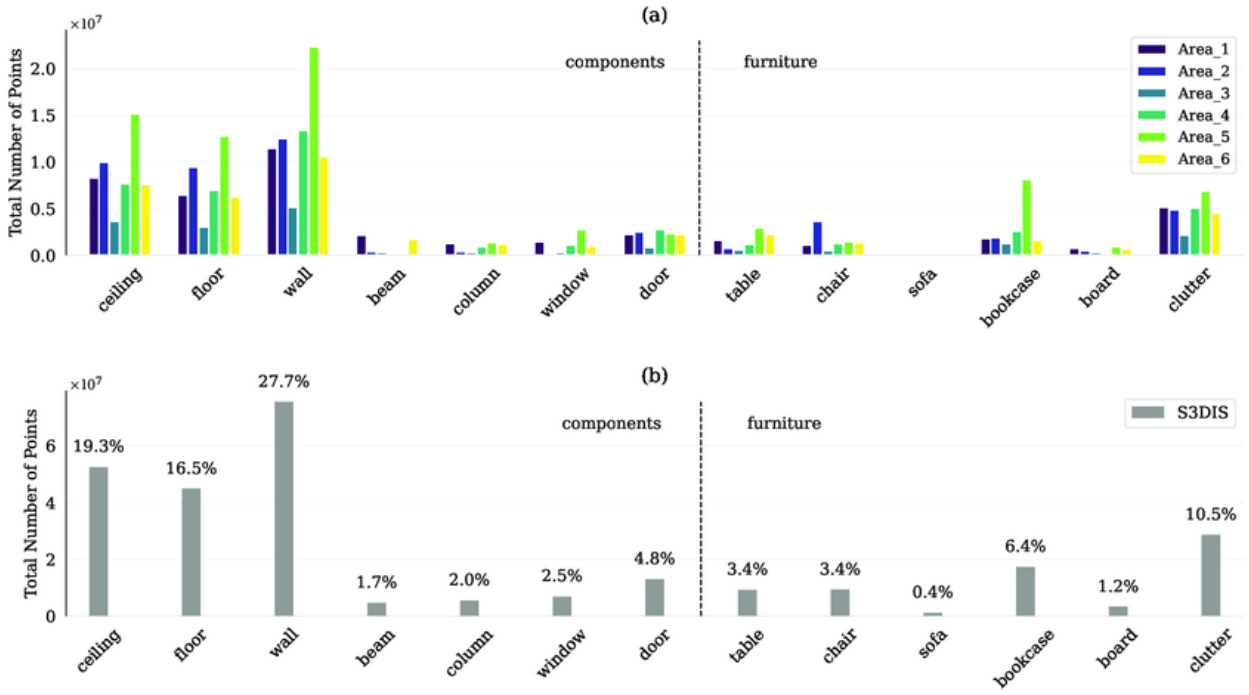

Ilustración 3.2: Distribución de clases de S3DIS por área y sobre el total

3.3. Dales

El Dayton Annotated Laser Earth Scan (DALES de ahora en adelante)[35] es un dataset público del año 2020 de escenas aéreas LiDAR dividido en 8 clases diferentes curadas por un equipo de expertos del organismo publicante, la Universidad de Dayton, Ohio.

Este conjunto de datos es el que el repositorio de la Pointnet++ original en MatLab[1] toma como base para entrenamiento y validación. En extensión cubre la misma cantidad de terreno que el anteriormente descrito, pero con una cantidad de puntos cercana pero inferior al medio millón. Esto lo convierte en un set de datos sobradamente atractivo para entrenamiento.

El conjunto de los puntos ha sido extraído utilizando un LiDAR modelo Riegl VQ-480 de doble canal [2] sobre un helicóptero que ha cubierto unos 330 kilómetros cuadrados en la ciudad de Surrey, Columbia Británica, Canadá.

Resulta relevante hablar de la densidad de los puntos: se encuentra en 20 por metro cuadrado, aunque, tras aplicar procesos de eliminación de ruido, ésta se queda unos 50 puntos por metro cuadrado.

Teniendo en cuenta que los ficheros serán divididos en celdas de 51x51m ($2601m^2$) vemos que tenemos información suficiente para que cada celda pueda contar sobradamente con los puntos de entrada de la red. En total contaríamos con unos 130.000 puntos por fichero de media.

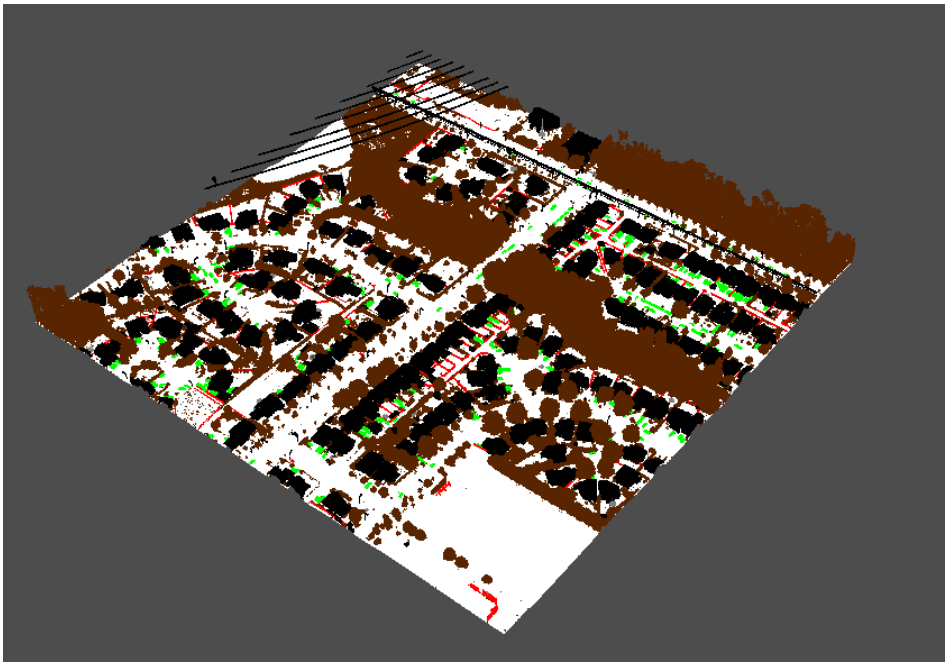

Ilustración 3.3: Representación 3D de una escena aérea de DALES

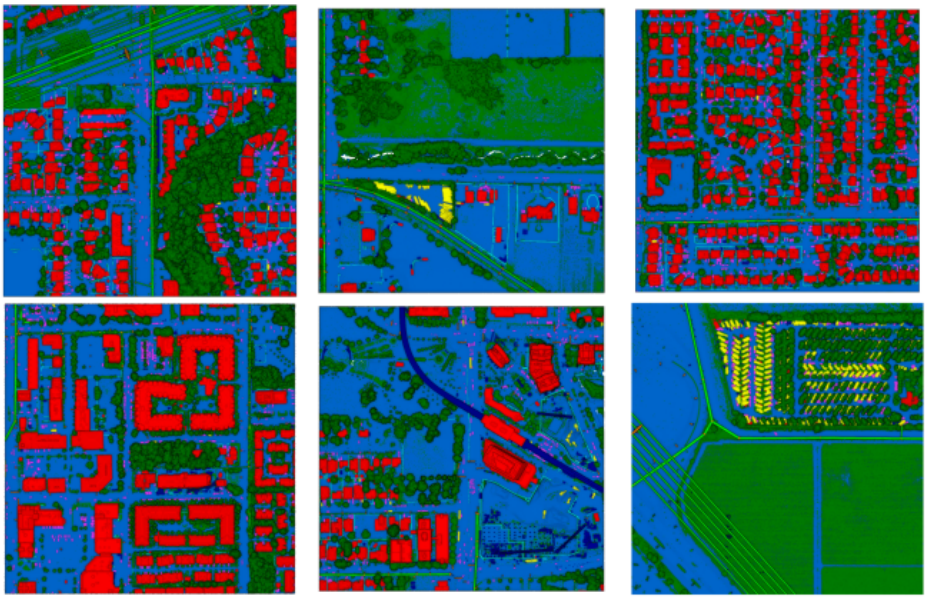

Ilustración 3.4: 6 escenas aéreas diferentes con las clases resaltadas a color. Cortesía de la University of Dayton

Este conjunto de datos presenta un balanceo razonable entre sus ocho clases para el tipo de escenas que representa, teniendo una predisposición lógica por las clases de suelo o vegetación (ver figura 3.5).

El número de puntos resulta más que suficiente para el tamaño esperado del modelo y la división en los split de entrenamiento y testeo solamente resulta ideal para el trabajo de preentrenamiento realizado en el sprint 5.

Estas ocho clases reflejan elementos urbanísticos típicos, siendo estos, en orden de cantidad de puntos; suelo, vegetación, edificios, desconocido, coches, vallas, cableado, camiones, torres eléctricas y farolas. Todas estas clases están presentes en la figura 3.4.

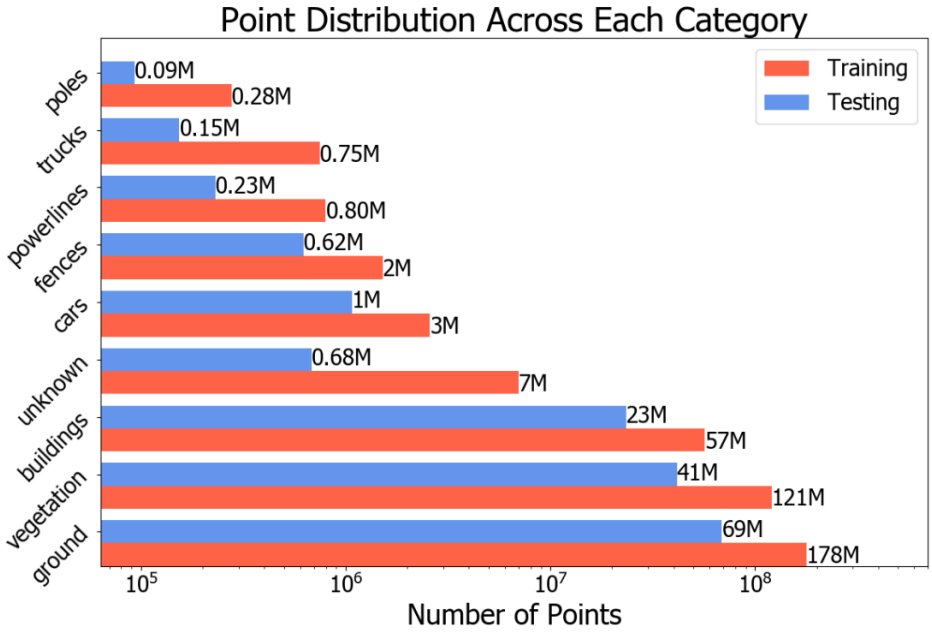

Ilustración 3.5: Distribución de clases dataset DALES. Cortesía de la University of Dayton

Pese a todo, sus mayores defectos radican en la simpleza del terreno que cubre, siendo una ciudad bastante más llana que la realidad de los datos de AeroLaser, y en la falta de puntos de las clases relativas al tendido eléctrico (powerlines y poles en la figura 3.5).

La página para encontrar el formulario y enlace de descarga de DALES se encuentra aquí.

3.4. ECLAIR

ECLAIR (Extended Classification of Lidar for AI Recognition) es un conjunto de datos de reciente publicación bajo el ala de la Aalto University, Finlandia.

Éste es un dataset diseñado específicamente para el entrenamiento y evaluación de modelos para la segmentación semántica de nubes de puntos y con ese fin han recolectado la, según sus autores, mayor cantidad de datos LiDAR en un solo dataset hasta la fecha[25]. Con $10km^2$ de extensión total y cerca de 600 millones de puntos repartidos en 11 clases diferentes, ECLAIR se alza como un conjunto ideal para el problema que nos atiene.

La recolección y filtrado de estos datos ha sido posible gracias a la combinación de un dispositivo LiDAR de largo alcance, una cámara RGB de alta resolución, cámaras hiperespectrales adicionales y sensores de temperatura y humedad montados en un helicóptero.

Gracias a esto se ha conseguido una densidad de puntos de primer orden que se encuentra alrededor de los 50 puntos por metro cuadrado[25] y como ya estableció en el apartado anterior sobre DALES, esto resulta más que suficiente para las necesidades de este proyecto.

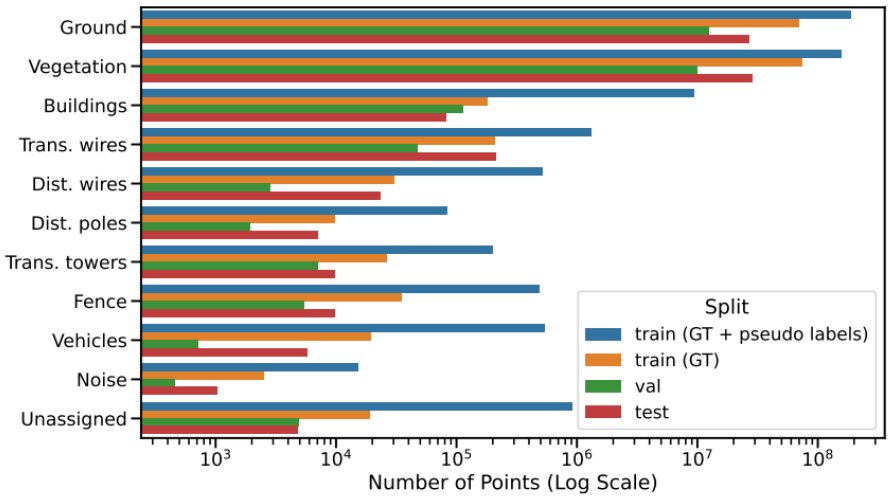

Ilustración 3.6: Distribución de clases ECLAIR. Figura cortesía de Aalto University

ECLAIR hace uso de una técnica avanzada propuesta por Dong-Hyun Lee [23] en el año 2013, llamada ”pseudo-labelling”. Esta técnica consiste en entrenar un modelo con datos ya etiquetados y de confianza y utilizar éste para enfrentarlo con una parte de un conjunto de datos no etiquetada.

La predicción generada pasa en ese instante a ser considerada la ground truth para esos puntos y se combina con los puntos anotados por una persona para generar un conjunto de datos mayor que el original

Éste es uno de los métodos llamados de aprendizaje semi-supervisados, esto es, subrogar una parte de la construcción de un conjunto de datos a un método que no requiera de

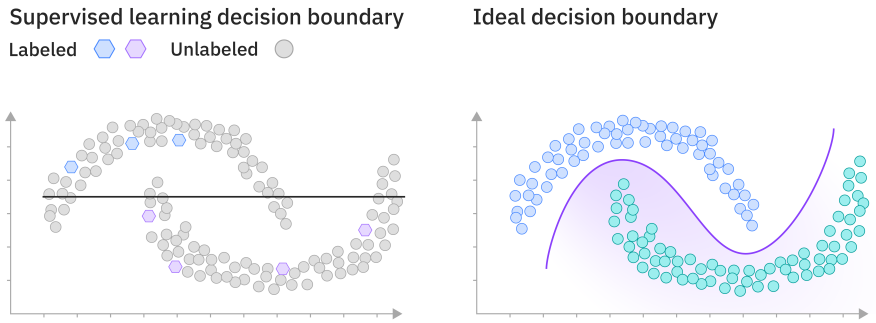

Ilustración 3.7: Ejemplo de funcionamiento ideal del pseudo-labelling. Cortesía de IBM

intervención humana (un modelo preentrenado en este caso) para reducir el costo humano y de tiempo.

Como el propio autor comenta en su paper[23], la mayor debilidad de este método es la baja cantidad de puntos de entrenamiento de la red. Esto se debe a lo sensible que es la anotación automática a la primera segmentación de todas, pudiendo sabotear todo el proceso si las primeras épocas de entrenamiento no tienen la calidad suficiente.

Por suerte, en este caso ECLAIR cuenta con puntos de sobra previo al pseudo-labelling (figura 3.6), por lo que esto no resulta problema alguno. De cara a los entrenamientos realizados con ECLAIR, todos han sido realizados utilizando los datos combinados con pseudo-labelling.

Como se puede observar en el esquema de clases de ECLAIR en la ilustración 3.6 (importante la escala logarítmica en el eje horizontal), este conjunto de datos presenta un desequilibrio de datos de gran magnitud si se ignora la técnica del pseudo-labelling utilizada en los ficheros de entrenamiento. Por este motivo se ha decidido utilizar los datos sometidos a esta técnica en el proyecto.

El enlace para el repositorio de GitHub de descarga de y prueba de ECLAIR se encuentra aquí.

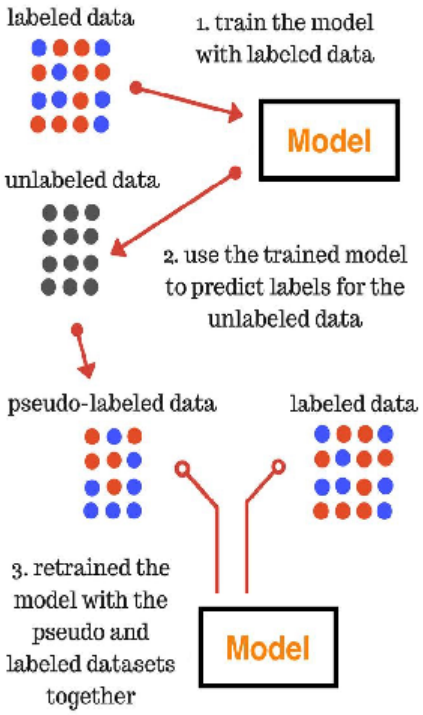

Ilustración 3.8: Esquema de funcionamiento del pseudolabeling. Cortesía de Medium

3.5. Aerolaser

Éste es el conjunto de datos troncal para el proyecto, puesto que es el que representa con más fidelidad los casos de uso a los que el modelo se tendría que enfrentar. Este conjunto de datos consta de unos 2 millones setecientos mil puntos distribuidos en 6 clases diferentes.

Ilustración 3.9: Sensor LiDAR propiedad de AeroLaser System S.L.

Estas clases corresponden, en orden numérico, al terreno o suelo, vegetación, edificios, torres eléctricas, cables y una clase llamada fondo, que recoge todo aquello que se salga de esta clasificación.

Ilustración 3.10: Sistema montado con LiDAR RIEGL VQ 480 II y cámaras. Cortesía de AeroLaser System S.L.

Como se puede observar en el cuadro 3.1, este conjunto de datos presenta una gran predilección por las clases de terreno y vegetación. Esto, pese a suponer un problema en el entrenamiento de un modelo balanceado, es un reflejo lógico de la realidad.

Observando los datos queda patente uno de los mayores retos de este dataset y es que, a diferencia de DALES, estos datos han sido tomados principalmente en zonas rurales. Esto provoca que la orografía sea más compleja que en ciudades y se favorezca la saturación de las clases de terreno y vegetación frente a la de edificación.

Estos datos son propiedad exclusiva de AeroLaser System S.L. y han sido cedidos para este proyecto específico. Todos los ficheros han sido anotados y revisados por personal de la empresa, garantizando su calidad y robustez.

Todas las escenas aéreas han sido obtenidas utilizando un LiDAR modelo RIEGL VQ 480 II (ilustración 3.10), montado sobre un helicóptero propiedad de la empresa. La ubicación concreta de las escenas es desconocida por motivos de protección de datos.

Formados en su totalidad por escenas aéreas del tendido eléctrico nacional (imagen 3.11), los ficheros de puntos han sido divididos siguiendo una proporción del 0,7 para el entrenamiento, 0,2 para el testeo y 0,1 para la validación.

Este dataset presenta un gran desbalanceo de clases marcado por una sobrerepresentación de la clase 0 o Terreno respecto a las demás, llegando ésta a presentar dos órdenes de magnitud

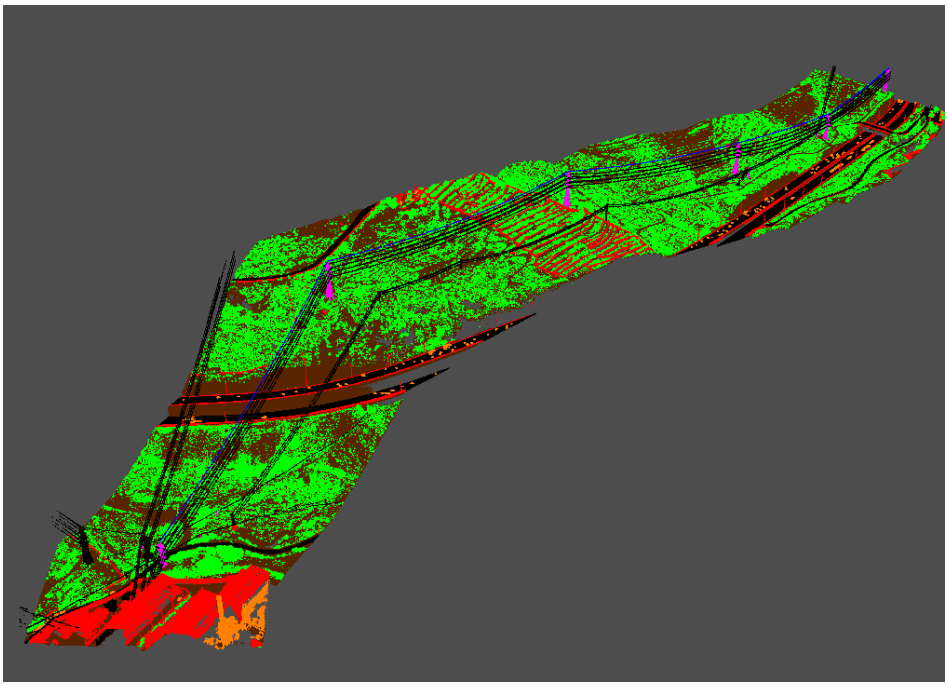

Ilustración 3.11: Escena aérea completa de Aerolaser System S.L.

por encima de las más escasas como se puede ver en el cuadro 3.1. Esto supondrá uno de los retos principales a superar en este proyecto y es motivo de decisiones de diseño tomadas más adelante.

	Terreno	Vegetación	Edificios	Torres eléctricas	Cables	Fondo
Entrenamiento	0,68231	0,29963	0,01406	0,00161	0,00166	0,00166
Validación	0,65439	0,32483	0,01661	0,00202	0,00147	0,00068
Test	0,69033	0,29582	0,00863	0,00101	0,00217	0,00204

Cuadro 3.1: Equilibrio de clases AeroLaser sobre la unidad

Capítulo 4

Desarrollo

4.1. Introducción

En este capítulo se detalla el proceso de creación del modelo final incluyendo las etapas de entrenamiento, validación y testeo, además de todo aquel trabajo necesario de recopilación y adaptación de los datos.

Esta sección ha sido divida en 6 sprints de duración variable, empezando por un sprint cero a modo de introducción a las herramientas y tecnologías usadas y con el objetivo de generar una base sólida para el óptimo crecimiento del programa. El resto de sprints se centra en el entrenamiento y afinado de los conjuntos de datos y la arquitectura utilizada hasta llegar al modelo final.

4.2. Sprint Cero

4.2.1. Familiarización con las herramientas a trabajar

Durante esta primera etapa se buscó la obtención de los conocimientos necesarios para lo que resta de proyecto. Esto incluye la lectura, modificación y escritura de datos en formatos LAS (usando la librería laspy[4]).

También se trabajó en el uso de las herramientas de trato de estructuras de datos mediante NumPy[6], la visualización de datos usando Open 3D[7], el uso de las herramientas básicas de PyTorch[8].

4.2.2. Diseño del código de preprocesado de datos

Debido al tamaño de los datos (sobre todo los proporcionados por Aerolaser), se hace necesario dar un tratamiento especial a los ficheros con el objetivo de disminuir la carga en memoria y reducir el número de etapas del entrenamiento. Este preprocesado ha de cumplir con unos objetivos concretos, que son, sin orden:

- ✔ Garantizar que los ficheros contengan al menos el mínimo número de puntos necesarios para la capa de entrada de la red.
- ✔ Asignar a los puntos las clases que les corresponden dentro del esquema del dataset. Esto es necesario en aquellos conjuntos de datos que tengan sus clases en el formato estándar del LiDAR[3].
- ✔ Asegurarse de que todos los puntos están clasificados entre 0 y n-1, siendo n el número de clases.
- ✔ Generar un fichero independiente por cada una de las celdas de 51x51m que salgan de cada escena aérea.
- ✔ Normalizar las tres coordenadas de todos los puntos en el rango [0,1].
- ✔ Escoger los puntos del fichero de manera aleatoria
- ✔ Dividir los ficheros entre entrenamiento, testeo y validación

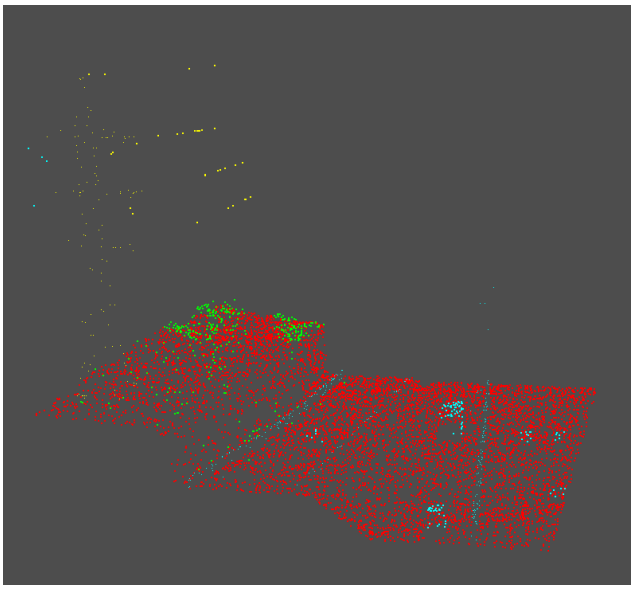

Ilustración 4.1: Celda del dataset de AeroLaser después del preprocesado

4.2.3. Visualización de los ficheros generados

Haciendo uso de la librería Open3D, es posible representar una nube de puntos en una escena tridimensional. Esto se realiza generando una geometría a partir de la nubes de pun-

tos, a continuación se voxeliza dicha geometría y se genera la visualización tridimensional[7] (ilustración 4.2).

De esta manera podemos comprobar que los datos se han almacenado correctamente en los ficheros, que la distribución de los puntos es aparentemente aleatoria y poder supervisar los ficheros que den algún problema durante el entrenamiento.

Ha de ser mencionado también un software de visualización de nubes de puntos proporcionado por la empresa AeroLaser System S.L. (imagen 4.3). Éste permite observar las nubes de puntos en mayor detalle que Open3D y la coloración de los puntos depende de la clase a la que pertenezca cada uno y puede ser modificada para reflejar aciertos/fallos.

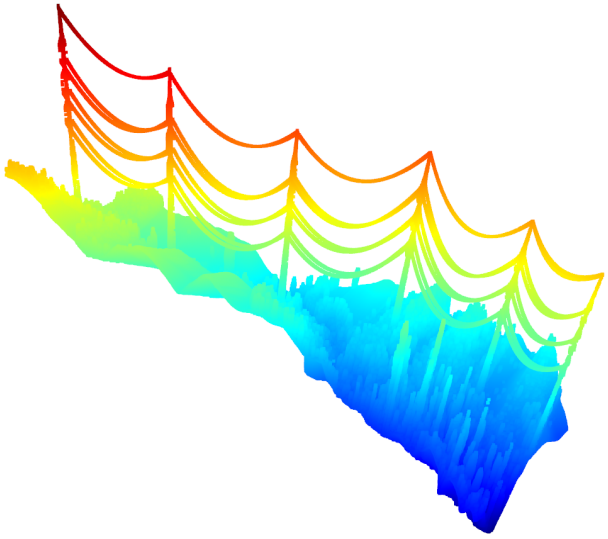

Ilustración 4.2: Fichero de AeroLaser visualizado con Open3D

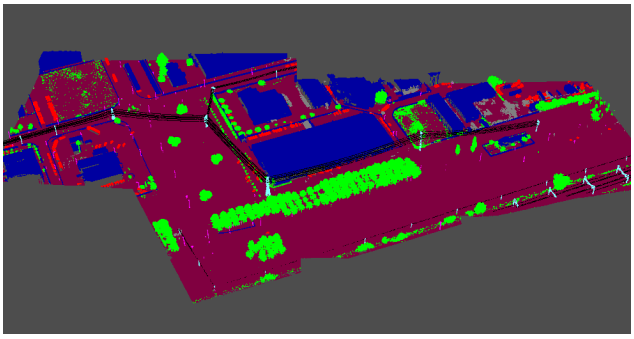

Ilustración 4.3: Fichero de AeroLaser visualizado con el software de AeroLaser

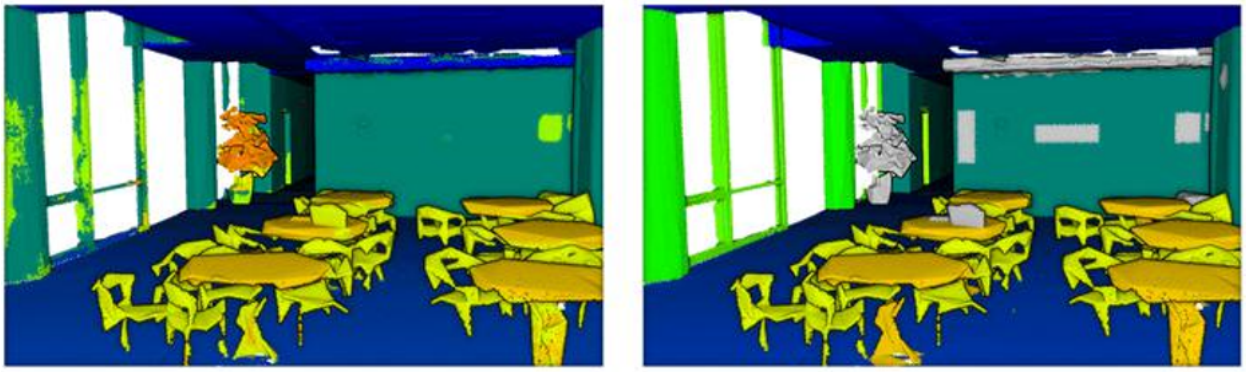

Ilustración 4.4: Detalle de una escena de S3DIS

4.2.4. Entrenar un modelo de prueba de la Pointnet++ con el dataset S3DIS

La Pointnet++ es una arquitectura que surge originalmente en MatLab y que tiene dos implementaciones sucesivas en Tensorflow y PyTorch. Por motivos de coherencia con el trabajo previo realizado en AeroLaser se tomó como base el repositorio de PyTorch[36]. Este repositorio usaba como ejemplo para su arquitectura de segmentación semántica el dataset DALES (comentado en el capítulo anterior).

En esta primera etapa de preparación y aprendizaje, se decidió comprobar la correcta funcionalidad del proyecto en PyTorch comparado con el repositorio original en MatLab. Para ello, se ejecutó el código en ambos lenguajes sin ninguna modificación, utilizando el mismo conjunto de datos.

El repositorio original no aporta más datos que la media de accuracy y de la IoU por lo que no se analizarán los datos obtenidos en detalle. Si cabe comentar que se han obtenido un 82.3 % de accuracy y una IoU de media del 51.09 %

Este modelo se comportó de manera muy similar a la esperada. Las diferencias en los datos obtenidos se pueden deber a factores tales como la aleatorización de los ficheros escogidos, diferencias de ejecución propias de usar lenguajes diferentes o al simple azar del aprendizaje de la red.

Con la funcionalidad comprobada y el proceso previo terminado es momento de empezar con los experimentos.

4.3. Sprint 1. Establecimiento de un punto de referencia mediante la obtención de resultados equiparables a los de la empresa colaboradora

La primera etapa de este sprint consistió en entrenar la nueva arquitectura [29] con los datos de AeroLaser, con el objeto de comprobar cómo de bien respondían estos. Los hiperparámetros tomados de referencia para este entrenamiento son aquellos por defecto en el repositorio de la Pointnet++ [36]. Esta decisión tiene como objetivo establecer un punto de partida del dataset sobre el que iterar y mejorar.

La figura 4.2 presenta un resumen de dichos hiperparámetros. Cabe destacar que no han sido utilizados todos los ficheros de entrenamiento, testeo o validación.

Cuadro 4.1: Hiperparámetros en el sprint 1

Número de puntos de entrada	1024
Épocas	32
Learning rate	0.001
Batch size	16
Ficheros de entrenamiento	4620
Ficheros de testeo	1320
Ficheros de validación	660
Tamaño de las celdas (m)	51x51

Si observamos los datos en la siguiente tabla, podemos ver que la red apenas tiene en cuenta las clases menos representadas. Esta falta de refuerzo llega incluso al extremo de no predecir ningún punto como miembro de las clases 4 y 5 correspondientes a los cables y a los elementos que no encajen en otra clase, respectivamente.

Estas métricas no resultan adecuadas para ningún tipo de modelo que pretenda segmentar nubes de puntos eficazmente. La mejor de las métricas (64 % de accuracy en la clase de terreno) sigue sin ser aceptable, por lo que queda patente la necesidad de una mejora en el proyecto. Esto puede hacerse, entre otras, aumentando el tamaño de la red. Esto se verá en el próximo sprint.

Esto se puede apreciar en la ilustración 4.5, donde los cables son clasificados como terreno por error.

Observando la distribución de clases del dataset, es fácil deducir por qué esta red, pensada para usarse con unas clases más balanceadas (figura 3.1), tendría este problema al entrenar. Este problema es el que se buscará resolver de aquí en adelante, haciendo uso de diversas técnicas y conjuntos de datos.

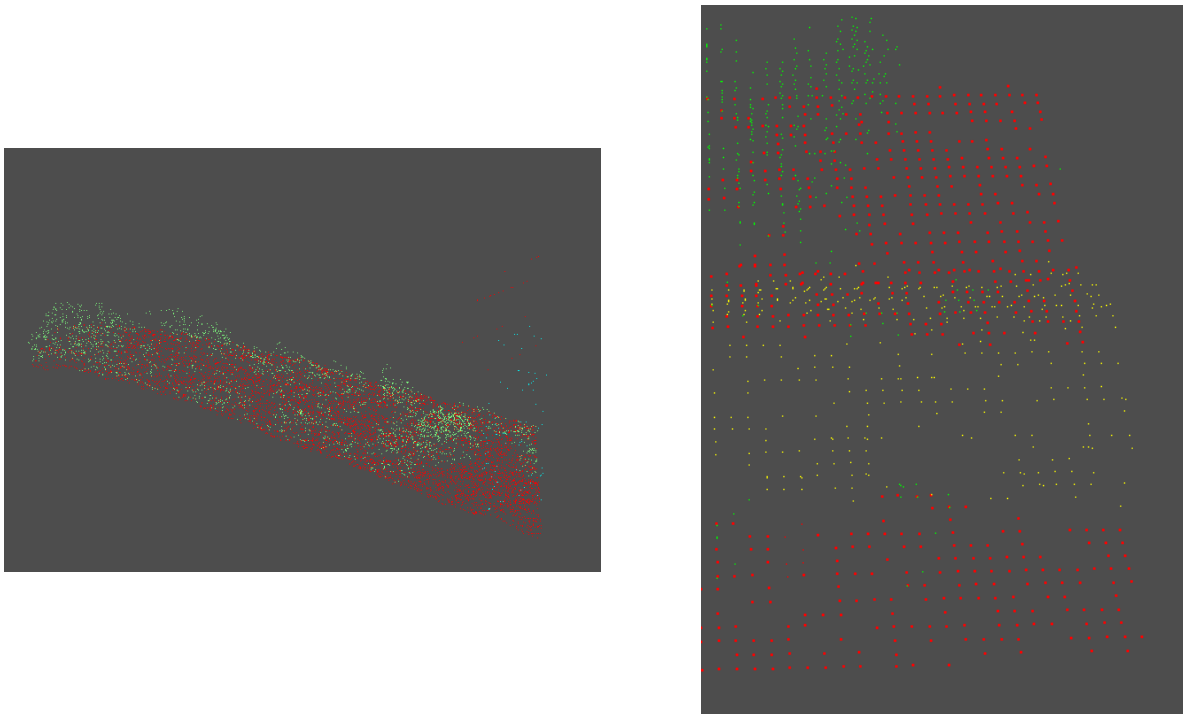

Ilustración 4.5: Ejemplo de cantidad de errores. Aciertos en verde, fallos en otros colores por claridad

Cuadro 4.2: Métricas de los datos de AeroLaser

Clases	Accuracy	IoU
0: Terreno	0.64	0.594
1: Vegetación	0.426	0.321
2: Edificios	0.54	0.048
3: Torres eléctricas	0.211	0.02
4: Cables	0	0
5: Fondo	0	0

4.4. Sprint 2. Aumento del tamaño de la arquitectura y pruebas con los hiperparámetros

Al abandonar los datos de DALES de manera temporal y adaptar el proyecto existente a los datos de AeroLaser System S.L., es necesario modificar el dataloader.

El dataloader es un programa esencial en el funcionamiento de este proyecto y es aquel encargado de suministrar a la red los puntos de la manera correcta. Resulta vital que este programa sea sólido, puesto que va a ser la base del resto de sprints.

El primer paso era adaptar el código planeado originalmente para trabajar con S3DIS al trato de fichero LAS utilizando laspy[4]. Gracias a esta librería se pueden manipular los ficheros a voluntad y extraer de estos información esencial.

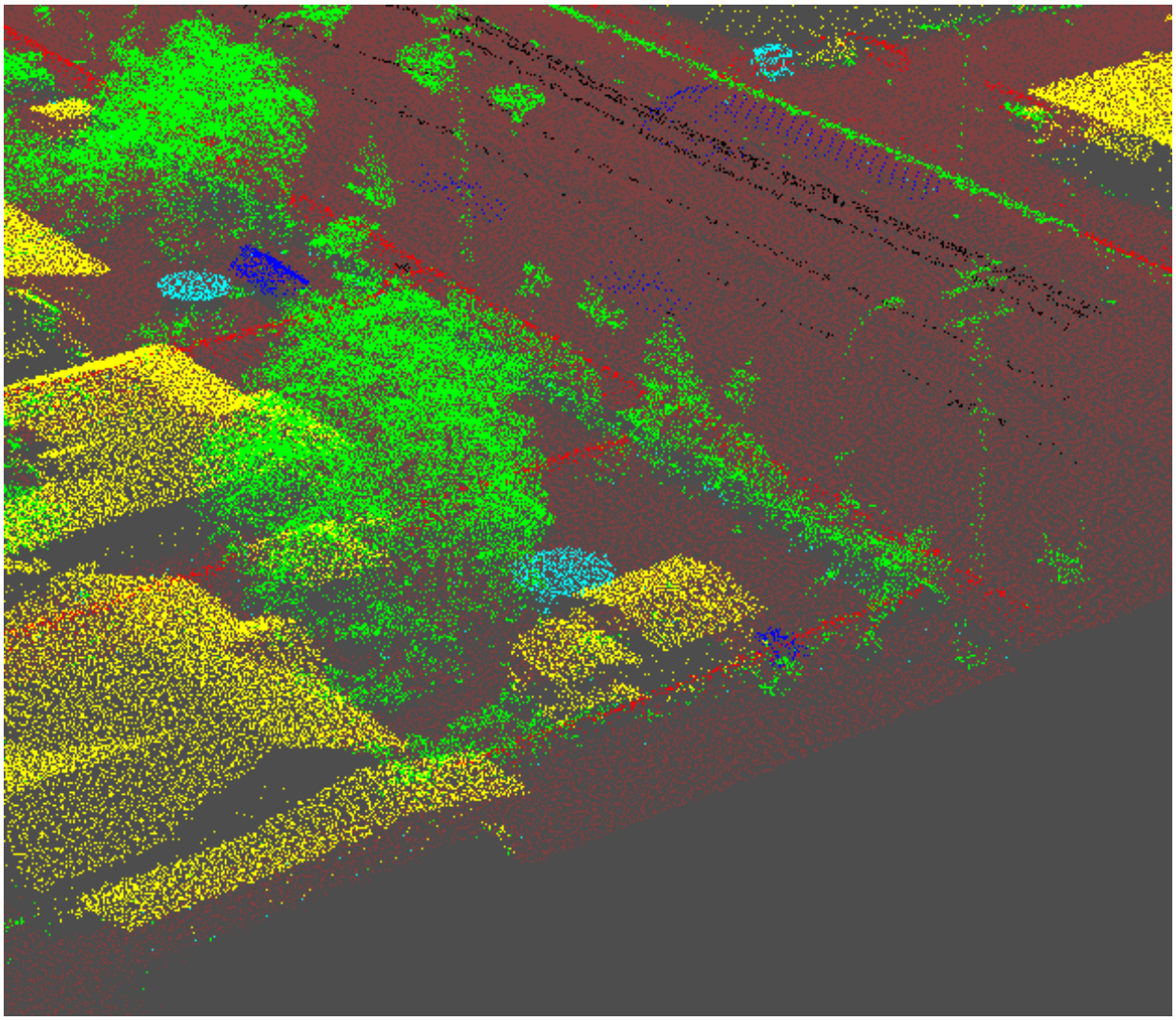

Ilustración 4.6: Ground Truth de una de las escenas utilizadas

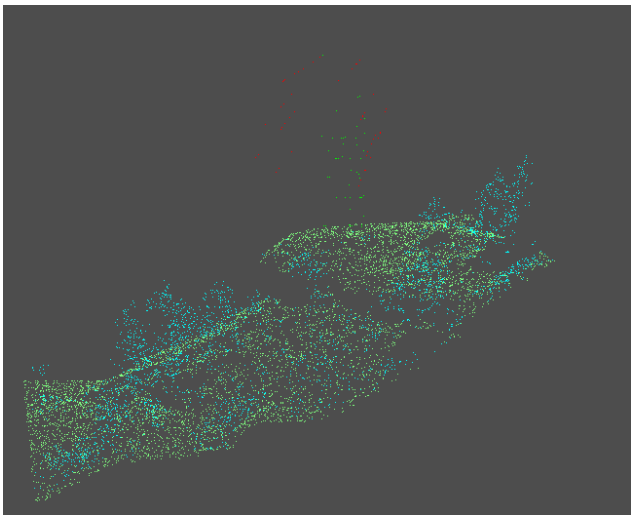

Ilustración 4.7: Celda del conjunto de AeroLaser después del preprocesado

En este punto toca recordar el código de preprocesado de datos ideado en el sprint cero. Este código tenía varios objetivos, entre los que estaba la normalización de la coordenadas en el rango de [0,1] para la correcta asimilación en la red y por anonimización de los lugares reflejados en las escenas.

Debido a las características del formato LAS las coordenadas georreferenciadas se almacenan generalmente en metros y se le aplica un offset o compensación y una escala para trasladarlo a la representación en kilómetros de la realidad.

Cuadro 4.3: Hiperparámetros en el sprint 2

Número de puntos	4096
Épocas	50
Learning rate	0.001
Batch size	16
Ficheros de entrenamiento	4620
Ficheros de testeo	1320
Ficheros de validación	660
Tamaño de las celdas (m)	51x51

Debido a esto, no es buena idea almacenar las coordenadas entre 0 y 1 en el formato de LAS sino guardarlas como vienen por defecto y realizar la normalización cuando se carguen los puntos en memoria.

Una vez los datos han sido extraídos y normalizados, lo que le resta al dataloader por hacer es obtener las clases correspondientes a cada punto y pasar toda esa información al programa principal para empezar el entrenamiento.

Todo el proceso previo resulta de un coste en memoria enorme debido a la densidad de información que contienen los ficheros LAS, especialmente los de AeroLaser. El procesado de toda esta información de manera simultánea ha sido posible gracias al uso exhaustivo e

intensivo de NumPy. Revisar la arquitectura de red era uno de los primeros pasos lógicos a

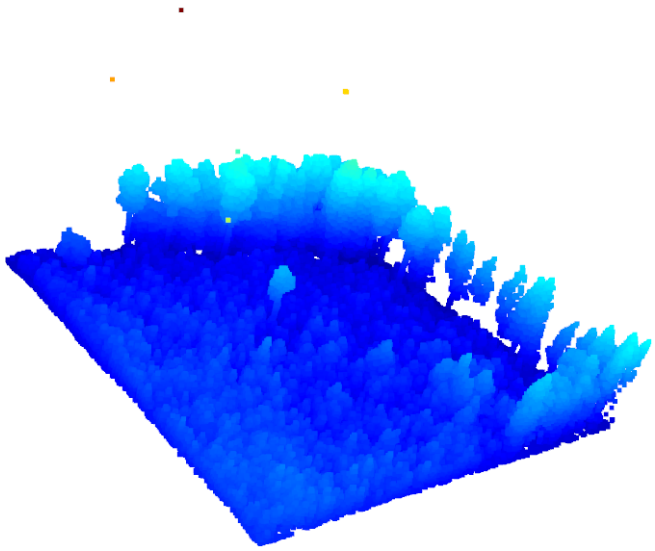

Ilustración 4.8: Celda solo con las clases de terreno y vegetación. Aquí se observa la densidad de información en cada fichero LAS

la hora de mejorar el modelo. Se decidió realizar una ampliación de la capa de entrada de la red pasando de 1024 a 4096 puntos y mantener en la misma medida el resto de capas. Con esto se busca respetar el funcionamiento propio de la Pointnet++, adaptándola para datos con mayor desequilibrio y variables que los de S3DIS.

Esto normalmente acarrearía riesgo de caer en el overfitting y reforzar aún más las clases más representadas, pero una de las grandes ventajas de la Pointnet (y, por ende, de la Pointnet++) es la capacidad de asimilar datos de tamaños y formas variables

Por otra parte, se consideró aumentar el número de épocas del entrenamiento y el tamaño de los batch. El objetivo de esta decisión era dar la oportunidad al modelo de reforzar las clases menos representadas y evitar un sobreentrenamiento a dos clases.

Tras el cambio realizado ,los hiperparámetros quedan como se observa en el cuadro 4.3

Los datos obtenidos tras entrenar con dichos hiperparámetros pueden verse reflejados en el cuadro 4.4

Si analizamos los datos de este sprint, vemos cómo son claramente mejores, aunque se quedan lejos de lo ideal. El problema de la no predicción de ciertas clases parece verse afectado de manera casi anecdótica por el incremento en la red.

A su vez, las clases más representadas sí ven aumentado cerca del 20 % su IoU y accuracy. Esta combinación de resultados nos lleva a plantear que la solución ha de pasar tanto por

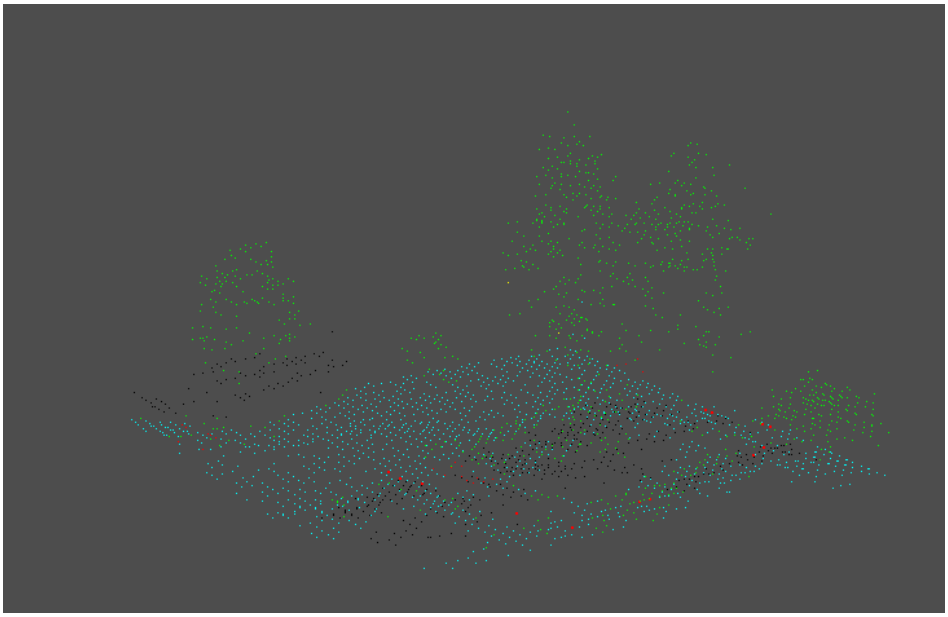

Ilustración 4.9: Segmentación de una celda con especial buen resultado. Los puntos rojos representan fallos.

Cuadro 4.4: Métricas de los datos de AeroLaser

Clases	Accuracy	IoU
0: Terreno	0.82	0.741
1: Vegetación	0.684	0.450
2: Edificios	0.32	0.339
3: Torres eléctricas	0.12	0.005
4: Cables	0.009	0.001
5: Fondo	0.017	0.001

aumentar el tamaño de la red como por buscar opciones alternativas. Esto se explorará en más detalle en la próxima sección.

4.5. Sprint 3. Entrenamiento con DALES y fine tuning

Tras entrenar con los datos de AeroLaser, era el momento de probar con el conjunto de datos DALES. Este dataset es alrededor del que se diseñó la Pointnet++ y por ende son de esperar mejores métricas.

Para el momento de la obtención de datos se decidió aprovechar la implementación original de la Pointnet++ para tener una base sobre la que comparar la versión modificada en PyTorch.

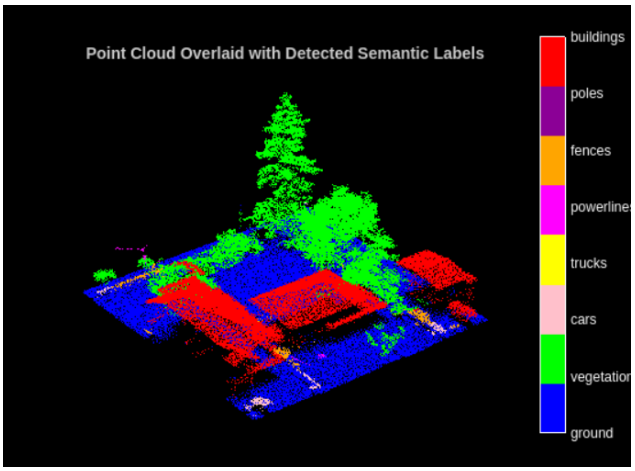

Ilustración 4.10: Ejemplo de celda segmentada en MatLab. Se puede apreciar la ausencia de la clase trucks en amarillo

Para que esta comparación sea lo más justa posible, se han realizado las pruebas con el tamaño original de la red. En este momento se ha observado que hay varios hiperparámetros por defecto diferentes entre la implementación de PyTorch y la original.

Hay que añadir que no ha sido modificado ninguno de los elementos del proyecto original en MatLab, pero que los datos suministrados a la red son aquellos pasados primero por el preprocesado diseñado en el sprint cero.

Entre estos hiperparámetros se encuentra el tamaño de la capa de entrada de la red que debería ser 8 veces más grande (de 1024 a 8192), el learning rate debería ser la mitad (0.001 a 0.0005) y el factor de descenso del learning rate (learning drop rate) a 0.1 desde 0.7.

El resumen de los hiperparámetros utilizados se puede ver en la figura 4.6. Cabe aclarar que el parámetro llamado learning period hace referencia al número de épocas en las que actualizar el learning rate.

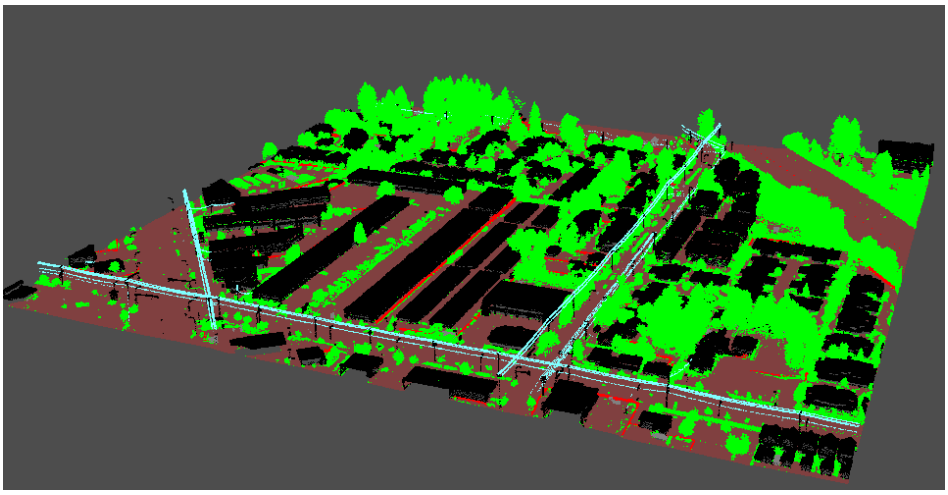

Ilustración 4.11: Ground Truth de una de las escenas usadas para testeo con DALES

Cuadro 4.5: Métricas comparativas entre MatLab y PyTorch

Clases	MatLab		PyTorch	
	Accuracy	IoU	Accuracy	IoU
Suelo	0.9771	0.9219	0.9888	0.95255
Vegetación	0.8341	0.8028	0.7989	0.78158
Coches	0.6351	0.3285	0.4683	0.27353
Camiones	0	0	0	0
Líneas eléctricas	0.7563	0.5413	0.7471	0.54610
Vallas	0.3231	0.1765	0.3398	0.14623
Postes	0.1429	0.0761	0.1257	0.05244
Edificios	0.9066	0.8107	0.9248	0.86249

En los datos plasmados en la tabla 4.5 podemos ver una mejora clara por encima de los datos previos del conjunto de AeroLaser. Pese a todo, llama la atención la ausencia de la clase camiones en los datos.

Esto en otros casos sería motivo de alarma, pero, en el problema que nos atiene, solo supone que, cuando establezcamos la relación entre las clases de AeroLaser y DALES (cuadro 4.7), la clase fondo estará algo menos representada.

Estas métricas abren la puerta al uso de este conjunto de datos, más que para comprobar el funcionamiento de la red. Esto se explorará en mayor profundidad en el próximo sprint

Viendo la mejor respuesta de los datos a esta nueva distribución de hiperparámetros, resultaría lógico utilizar estos de aquí en adelante para en entrenamiento general de la red. Este crecimiento conlleva una serie de problemas asociados que habrán de ser resueltos próximamente.

El esquema final de la red e hiperparámetros es el que podemos observar en la figura4.6.

Cuadro 4.6: Hiperparámetros en el sprint 3

Número de puntos	8192
Épocas	20
Learning rate	0.0005
Learning drop rate	0.1
Learning period	10
Batch size	16
Ficheros de entrenamiento	3280
Ficheros de testeo	937
Ficheros de validación	468

El resultado satisfactorio de este entrenamiento llevó a la idea de combinar los conjuntos de datos. En primera etapa, entrenar la red con un conjunto de datos numeroso, balanceado y de eficacia probada, combinado con un segundo entrenamiento con los datos de AeroLaser para reforzar las clases menos balanceadas.

4.6. Sprint 4. Entrenamiento combinado con DALES y AeroLaser

Tras comprobar el buen funcionamiento de los datos de DALES en las clases de mayor relevancia, era momento de probar una opción nueva inspirada en el paper comparativo entre KPConv y la Pointnet[31].

En el sprint 2 quedó patente que la mayor debilidad del conjunto de AeroLaser radicaba en las clases que se salieran de las dos más significativas. En cambio, el conjunto de DALES presenta mayor equilibrio que éste y ya ha obtenido mayores resultados por defecto.

Teniendo esto en cuenta, se abre la opción de aprovechar el modelo final del sprint anterior y hacerlo pasar por un proceso de reentreno con los datos de AeroLaser. Con esto se conseguiría un modelo adaptado a la orografía compleja manteniendo unas métricas aceptables para todas las clases de manera transversal.

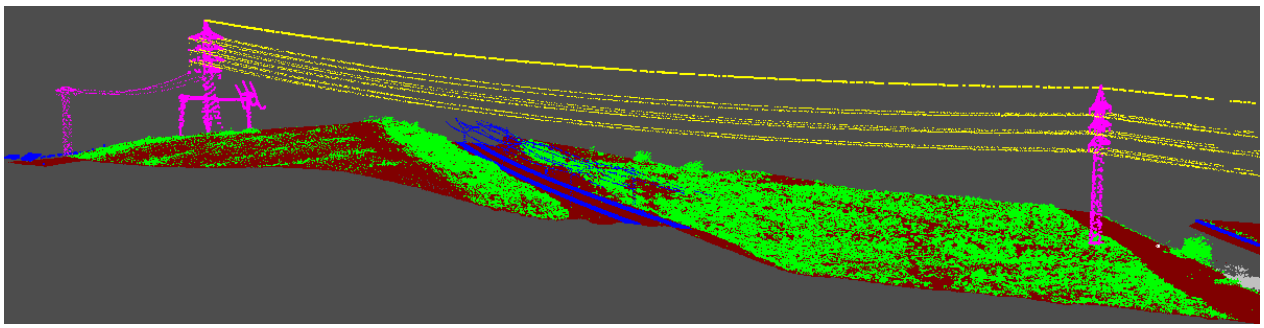

Ilustración 4.12: Ground truth de una de las escenas aéreas antes del preprocesado

Por desgracia no es posible, al menos cuando se utiliza PyTorch, volver a entrenar un modelo con una cantidad diferente de clases. La solución a esto pasa por modificar el preprocesado de datos y la segmentación.

La modificación del software de preprocesado resultaba necesaria por varios motivos. El primero y más evidente es que, al aumentar el tamaño de entrada de la red, los ficheros con 4096 puntos no eran suficientes.

Además fue necesario adaptar los códigos de las ocho clases de DALES a las seis clases de AeroLaser. La relación entre las clases antiguas y las nuevas queda plasmada en el cuadro 4.7

Una vez reconstruidos los ficheros de entrenamiento, aparece un problema: los datos de DALES son con diferencia más ligeros que los de AeroLaser y, al llevarlos al rango de los 8192 puntos de entrada, ya forzaban la capacidad del equipo.

Por esto se decidió realizar una diferencia en la carga de datos del código. En este momento, el programa aloja todos los ficheros de entrenamiento en memoria y los lee uno por uno en cada momento.

Cuadro 4.7: Métricas finales Sprint 4

Clase DALES	Clase AeroLaser
1: Terreno	0: Terreno
2: Vegetación	1: Vegetación
3: Coches	5: Fondo
4 : Camiones	5: Fondo
5: Cables	4: Cables
6: Vallas	2: Edificios
7: Torres eléctricas	3: Torres eléctricas
8: Edificios	2:Edificios

Esta manera de funcionar es altamente ineficaz, especialmente teniendo en cuenta la densidad de información en los ficheros. Por ello, a partir de este momento, el código del dataloader lee los ficheros uno a uno desde el disco y se queda solo con la información esencial (coordenadas y clase).

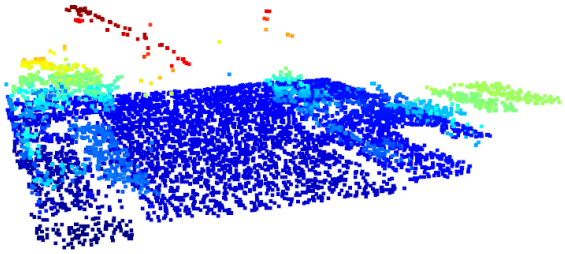

Ilustración 4.13: Una de las celdas de entrenamiento de DALES. Apréciese la variedad de los puntos escogidos aleatoriamente

Además, se le añade al método del dataloader, la responsabilidad de actuar como garbage collector (programa encargado de eliminar de memoria los datos que hayan quedado inútiles) para intentar suplir una de las mayores limitaciones de entrenar en Python.

Una vez modificado el preprocesado y el dataloader solo queda entrenar primero con DALES y luego realizar el transfer learning con AeroLaser.

Los hiperparámetros de entrenamiento son exactamente los mismos que los utilizados en el cuadro 4.6, con la diferencia de que, para la segunda parte del entrenamiento, se bajó el learning period de 10 a 5 para dar tiempo a la red de asimilar los cambios entre ambos conjuntos.

Tras terminar el entrenamiento y el transfer learning, las métricas son las que figuran en la tabla 4.8. Observando los datos finales, podemos concluir que este método parece, si no la opción correcta, la mejor hasta el momento.

Cuadro 4.8: Relación de clases entre DALES y AeroLaser

Clases	Accuracy	IoU
0: Terreno	0.9618	0.9388
1: Vegetación	0.79523	0.73108
2: Edificios	0.81459	0.78756
3: Torres eléctricas	0.009	0.00526
4: Cables	0.34333	0.3263
5: Fondo	0.0061	0.0044

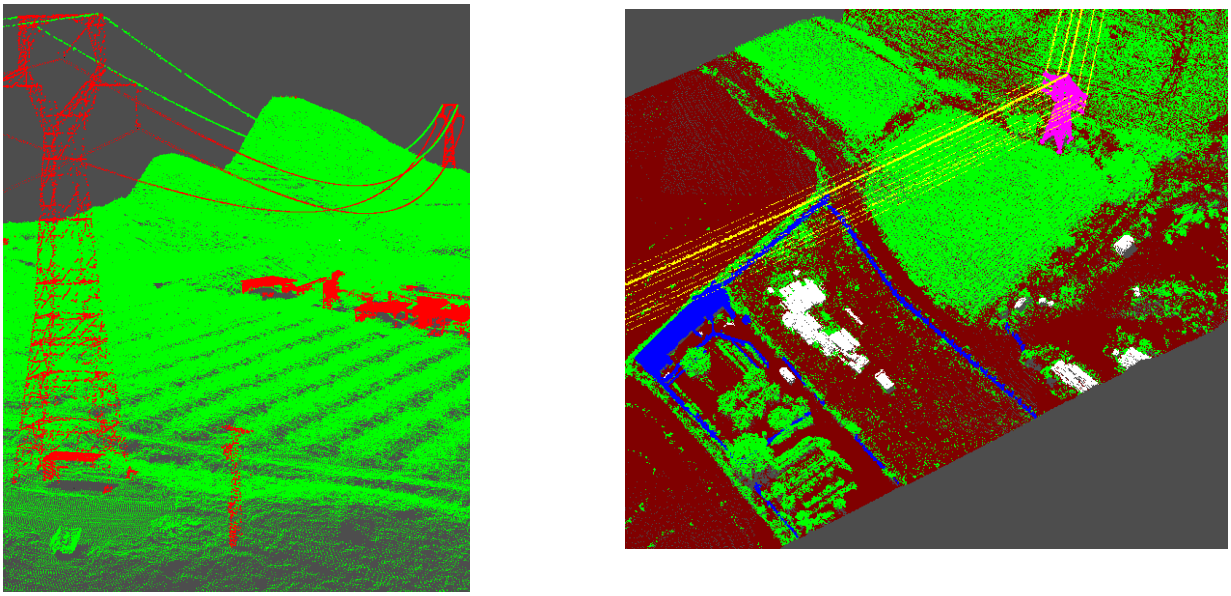

Ilustración 4.14: Comparativa entre ground truth y aciertos/fallos antes del transfer learning

Podemos ver cómo la clase de terreno ya llega a métricas más que aceptables y que las de vegetación y edificios se quedan cercanas a los mejores datos en la iteración anterior.

De hecho, es posible, observar al comparar las ilustraciones 4.14 y 4.16, que el entrenamiento con AeroLaser permite a la red segmentar zonas en las que antes fallaba.

Pese a la mejora, parece que las clases de torres eléctricas y fondo se resisten a ser identificadas con un mínimo de éxito. Para paliar esto, habrá que buscar una alternativa al crecimiento de la red.

En este momento del desarrollo, el entrenamiento de 20 épocas de AeroLaser lleva algo más de 5 días. Haciendo unos cálculos rápidos, entrenar la red con 50 épocas se tardarían más o menos doce días y medio.

Esto, en caso de caer en el overfitting o que no resultase la opción correcta, pondría el proyecto en una situación demasiado precaria, alargando demasiado los tiempos de entrenamiento.

Es por esto que se decidió que el siguiente paso era repetir el éxito de este sprint, pero

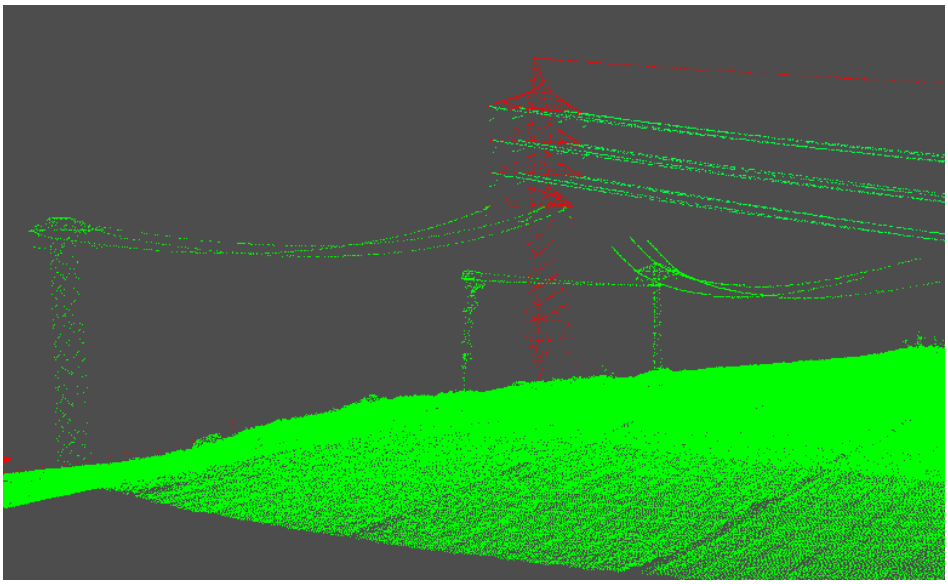

Ilustración 4.15: Detalle de los aciertos y fallos, donde se aprecia la dificultad de la clase Torre eléctrica. Ver Ilustración 4.12

realizando el primer entrenamiento con un conjunto de datos a priori mejor construido y con mayor similitud al conjunto de AeroLaser.

Con esto se da por concluido el sprint 4 y pasamos al quinto y último sprint con el proceso de transfer learning con ECLAIR como base.

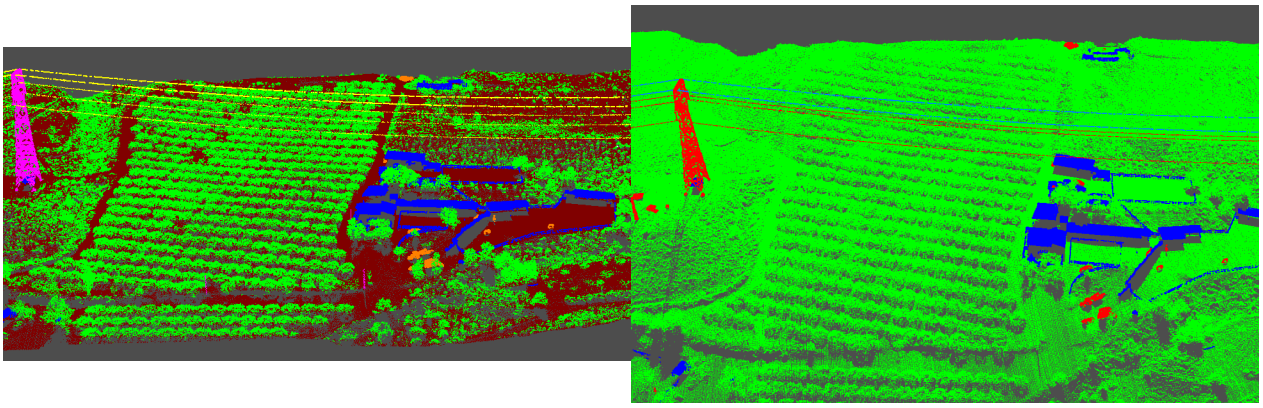

Ilustración 4.16: Comparativa entre ground truth y aciertos/fallos. Los fallos en rojo, los aciertos usan varios colores para más claridad

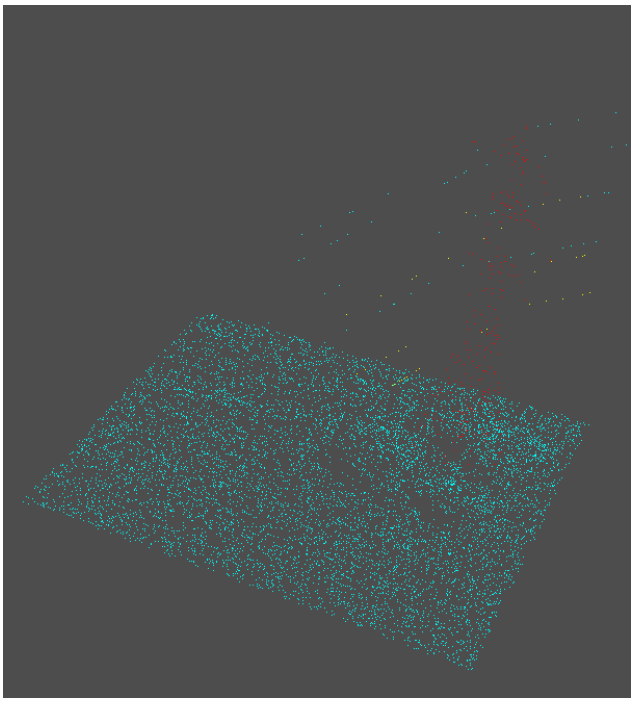

Ilustración 4.17: Segmentación de una celda de AeroLaser. Se puede apreciar cómo buena parte de los cables se clasifican como terreno

4.7. Sprint 5 Entrenamiento combinado con ECLAIR y AeroLaser

Para este sprint se decidió realizar un experimento similar al anterior, pero con el conjunto de datos ECLAIR.

Como se comentó con anterioridad, en el capítulo 3.4, este dataset posee una cantidad y variedad de datos mayor que el DALES. Por esto, sería lógico inferir que éste debería devolver resultados mejores en el entrenamiento que el sprint anterior, aunque esto tendrá que ser demostrado empíricamente.

En esta parte podemos recuperar el código de preprocesado del sprint anterior con ligeras diferencias. En este caso, los ficheros de ECLAIR están en el formato comprimido LAS Zip (LAZ como extensión de archivo), por lo que habrán de ser transformados al formato LAS tradicional. Además, el cambio de clases debido al uso de doce clases por parte de ECLAIR con respecto a las seis de DALES (cuadro 4.9).

En este caso se han usado los mismos hiperparámetros que en el sprint anterior con el objetivo de comparar ambos datasets en igualdad de condiciones.

Tras un tiempo levemente mayor de entrenamiento en la primera etapa y similar en la segunda, se han obtenido las métricas finales de este proceso. Éstas están reflejadas en la tabla 4.10

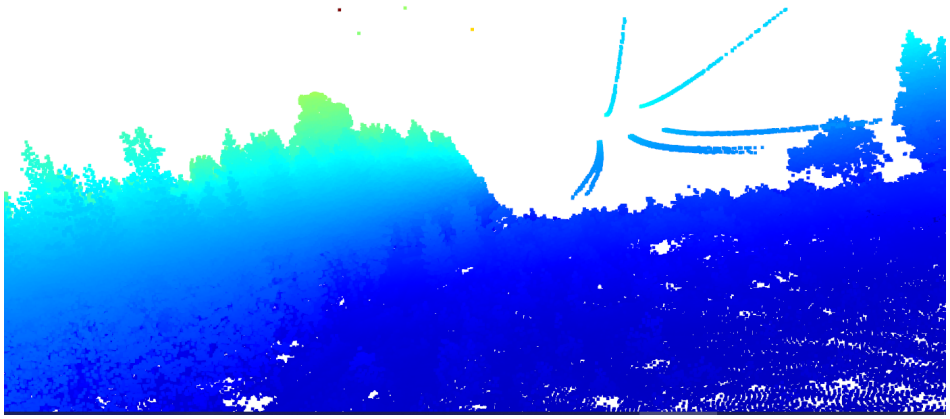

Ilustración 4.18: Detalle de escena de ECLAIR usada en entrenamiento. Llaman la atención los puntos flotantes, correspondientes a la categoría de ruido

Cuadro 4.9: Relación de clases entre ECLAIR y AeroLaser

Clase DALES	Clase AeroLaser
2: Terreno	0: Terreno
3: Vegetación	1: Vegetación
4: Edificios	2:Edificios
5 : Ruido	5: Fondo
6: Cables alta tensión	4: Cables
7: Cables baja tensión	4: Cables
8: Torres baja tensión	3: Torres eléctricas
9: Torres alta tensión	3: Torres eléctricas
10: Vallas	2:Edificios
11: Vehículos	5: Fondo
1: No asignado	5: Fondo

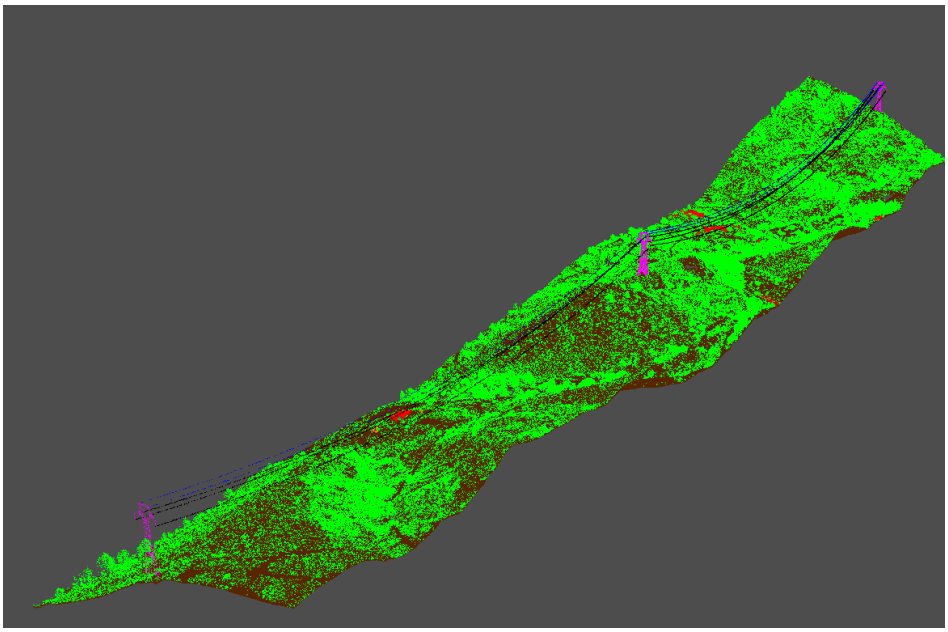

Ilustración 4.19: Ejemplo de ground truth de una de las escenas usadas antes del preprocesado.

Si comparamos estos resultados con los obtenidos en el sprint anterior (cuadro 4.8), observamos cómo las métricas son esencialmente diferentes a las obtenidas con DALES.

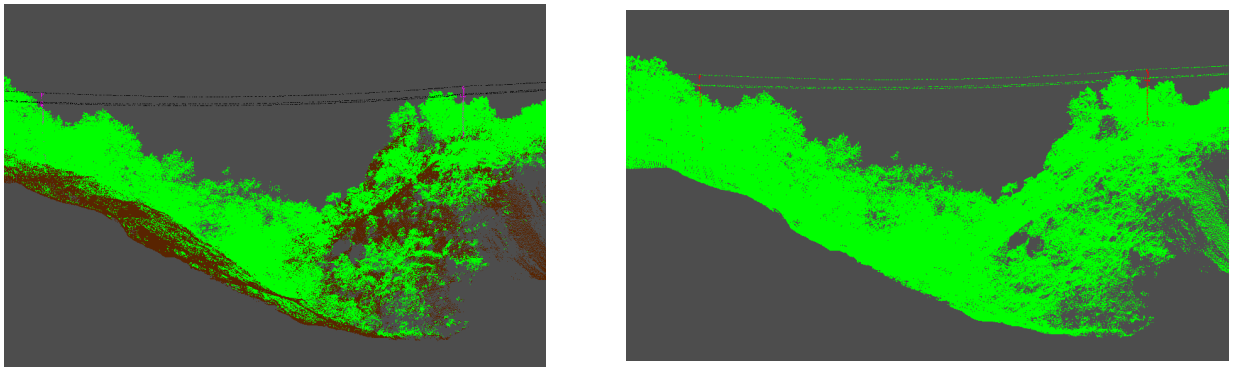

Ilustración 4.20: Comparativa entre ground truth y aciertos/fallos. Los fallos en rojo, los aciertos en verde

Se observa cómo el modelo entrenado con el conjunto de la University of Dayton, Ohio potencia fuertemente las clases más representadas. Por su parte, el entrenamiento con ECLAIR se comporta peor con éstas pero otorga mejores resultados, por lo general, en las carencias del anterior modelo.

Pese a esto, se observa cómo la clase correspondiente a torres eléctricas sufre gravemente el desbalanceo de clases y se resiente en mayor medida que con DALES. Esto es especialmente evidente en la comparativa de la figura 4.20.

Cuadro 4.10: Métricas finales Sprint 5

Clases	Accuracy	IoU
0: Terreno	0.84381	0.82608
1: Vegetación	0.7139	0.70383
2: Edificios	0.53623	0.50759
4 Torres eléctricas	0.0048	0.0043
5 Cables	0.72654	0.70396
6 Fondo	0.71667	0.68254

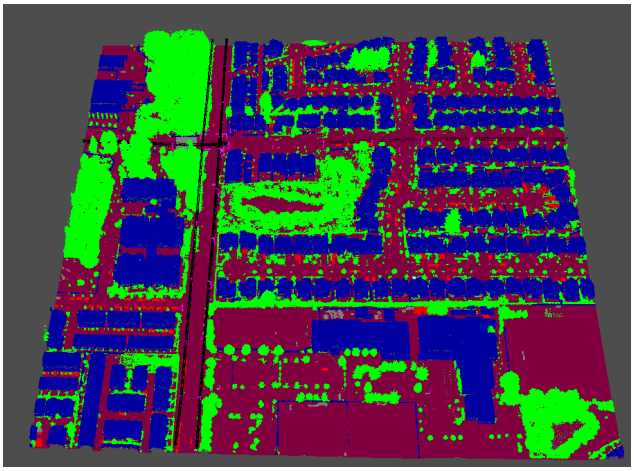

Ilustración 4.21: Ground truth de una de las escenas de entrenamiento

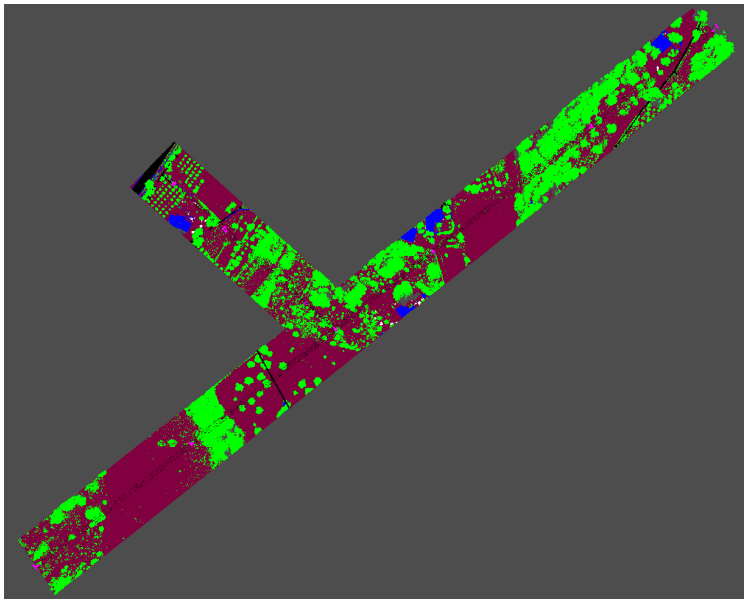

Ilustración 4.22: Ground Truth de una de las escenas de validación de AeroLaser

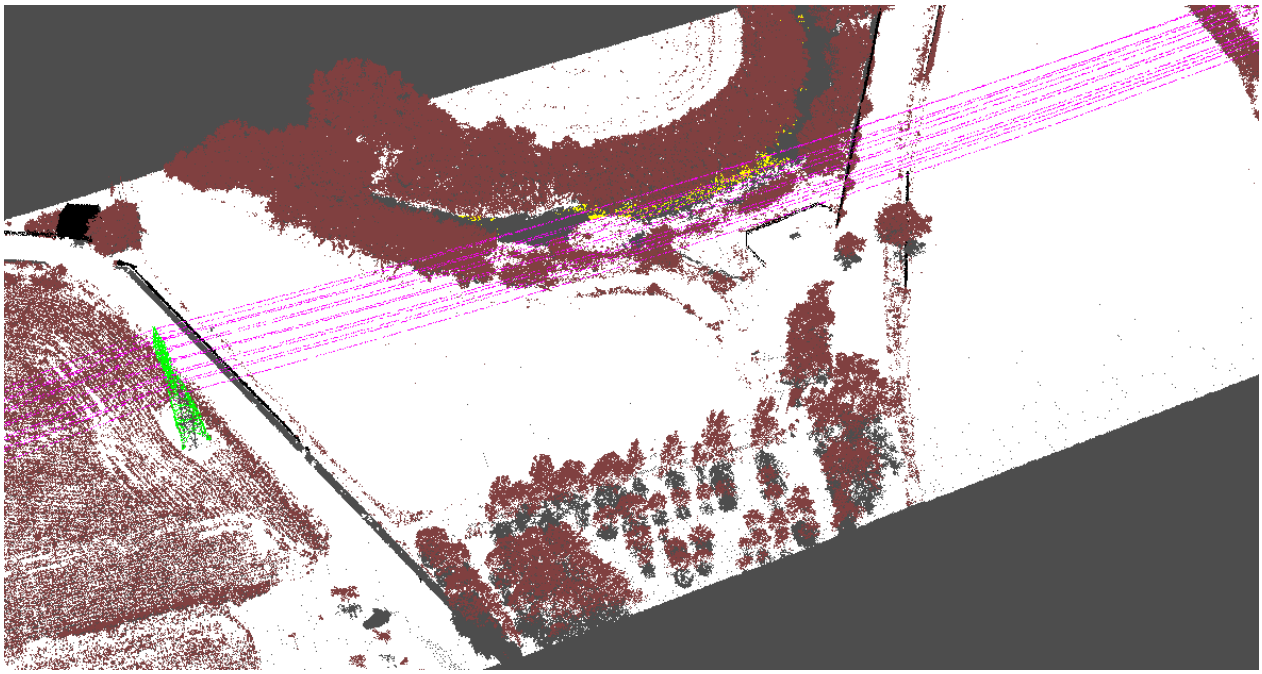

Ilustración 4.23: Segmento de una escena de ECLAIR

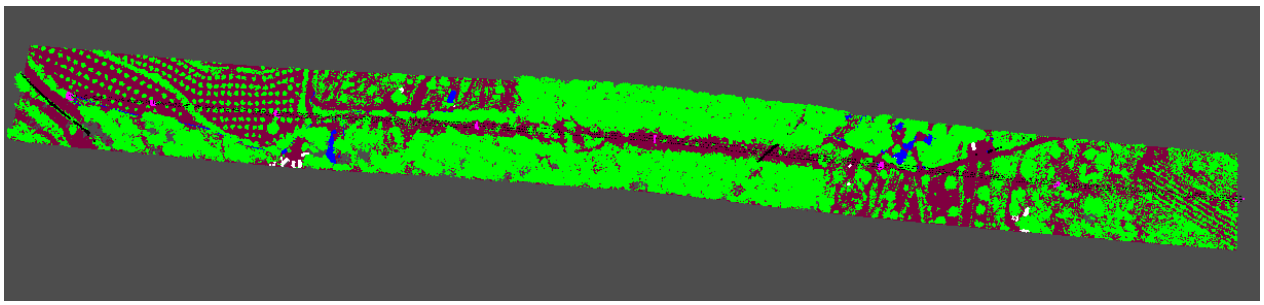

Ilustración 4.24: Ground Truth de una de las escenas de validación. Nótese la baja cantidad de puntos de la clase torre eléctrica (rosa)

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

Tras todo el trabajo, análisis del estado del arte y experimentación, queda patente que, de entre todas las opciones barajadas, ninguna de las dos opciones finales resulta claramente superior.

Las diferencias entre los modelos pre entrenados radica en la potencia de DALES con las clases más representadas, frente a la regularidad de ECLAIR entre todas.

Esto se puede apreciar especialmente en la tabla comparativa 5.1, donde vemos cómo, en la clase más numerosa DALES, llega a lograr una ventaja del 9 % en la IoU. Ventaja que en menor medida posee en las clases vegetación y edificios.

Cuadro 5.1: Métricas comparativas del transfer learning entre DALES y ECLAIR

Clases	DALES		ECLAIR	
	Accuracy	IoU	Accuracy	IoU
0: Terreno	**0.9618**	**0.9388**	0.84381	0.82608
1: Vegetación	**0.79523**	**0.73108**	0.7139	0.70383
2: Edificios	**0.81459**	**0.78756**	0.53623	0.50759
3: Torres eléctricas	0.009	0.00526	0.0048	0.0043
4: Cables	0.34333	0.3263	**0.72654**	**0.70396**
5: Fondo	0.0061	0.0044	**0.71667**	**0.68254**

A su vez, ECLAIR logra alzarse por encima de Dales con un 40 % de ventaja en ambas métricas en la clase de cables y con un sorprendente 70 % en la clase de fondo.

Las diferencias en la clase de torres eléctricas resultan tan insignificantes y en ambos casos

aportan datos tan bajos que es seguro afirmar que las predicciones de esas clases en ambos modelos son fortuitas.

Esta diferencia entre clases resulta natural en buena parte si tenemos en cuenta la realidad de DALES y de ECLAIR. El primero de los datasets representa escenas casi enteramente urbanas, mientras que en el caso de ECLAIR, éste representa un conjunto mixto basado en la realidad que no busca una repartición equitativa de sus puntos. Citando el paper de ECLAIR [25] : "This imbalance reflects real-world conditions, presenting an opportunity to test the robustness and generalizability of point cloud classification models under skewed distribution scenarios."

A su vez, resulta natural que una red que tiene una sobrerepresentación de puntos de ciertas clases se centre en predicciones de esas pocas, frente a una red más balanceada en la que los aciertos se reparten y los fallos también.

Si prestamos atención a las medias aritméticas de los datos, podemos observar cómo la primera versión ECLAIR prácticamente dobla los datos del entrenamiento aislado de AeroLaser en la mejor de sus versiones.

Cuadro 5.2: Métricas finales Sprint 5

	Media Accuracy	Media IoU
AeroLaser	0.3028	0.2458
DALES y AeroLaser	0.4883	0.4735
ECLAIR y AeroLaser	0.5903	0.5714

Tras analizar todas las métricas presentes, llegamos a varias conclusiones. La primera es que el conjunto de datos de AeroLaser necesita de apoyos si se pretende entrenar un modelo especializado. La mejora del dataset puede pasar por varias técnicas, que serán explicadas más adelante.

La otra de las conclusiones lógicas es que ambos modelos presentan fortalezas principales evidentes. Por esto, ambos métodos deberían ser viables para según las necesidades de cada problema.

En caso de presentarse la necesidad del uso de un modelo especializado en la distinción de vegetación sobre edificaciones, como podría ser el caso de una red encargada del control automatizado de incendios, el conjunto de datos de DALES es la respuesta evidente en este caso.

Si, por el contrario, se necesitase de un modelo capaz de adaptarse a una mayor variedad de clases y a un tipo de escenas de geometría más compleja, la opción de ECLAIR resulta mejor alternativa.

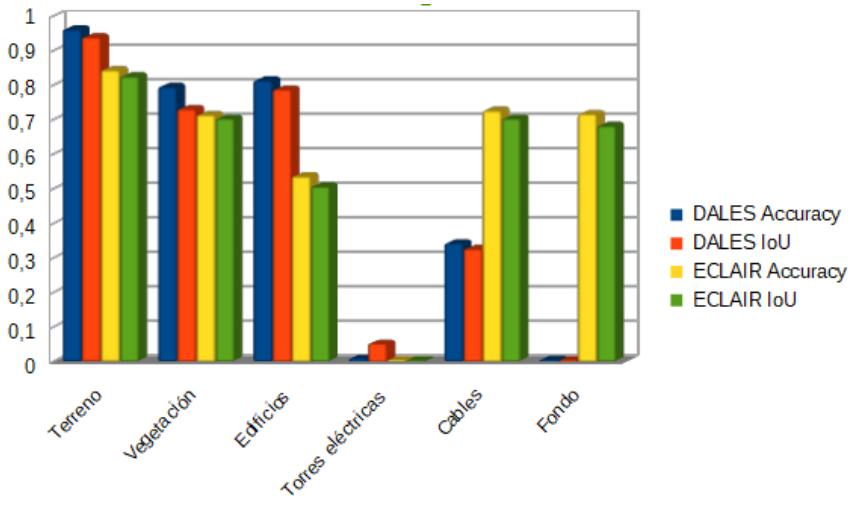

Ilustración 5.1: Gráfico de los datos del cuadro 5.1

5.2. Trabajo futuro

De cara a una posible ampliación del trabajo, éste podría tomar diferentes rutas. La primera de ellas sería la de realizar experimentos semejantes sobre KPConv. Como ya se ha establecido previamente, ésta se queda escasamente por detrás de la Pointnet++, lo cual lleva a plantearla como otra opción viable si los datos así lo plasmasen para este problema.

Cabe destacar que obtener los mejores resultados de KPConv es computacionalmente costoso debido al uso de los kernels dinámicos y sin ello probablemente se encontraría por detrás de los modelos aquí desarrollados.

Esto resulta de interés al ver que ya el tamaño de la Pointnet++, con los datos utilizados supone un reto para un equipo de máximo nivel como es el aportado por la ULPGC. Debido a esto, los experimentos con KPConv han de realizarse bajo el conocimiento de que la mejora no está garantizada y que el tiempo de entrenamiento sería mucho mayor.

La siguiente de las opciones sería la construcción de un conjunto de datos más balanceado y con énfasis en la clase de torres eléctricas. Esto resulta complejo debido al hecho de que los ficheros representan escenas de terreno reales, pero podría resultar factible.

La mejora del conjunto de datos de AeroLaser no es trivial y las posibles maneras de lograrlo serán exploradas a continuación

5.2.1. Mejora del conjunto de datos

Una de las técnicas para lograr esto sería mediante un proceso de data augmentation aplicado exclusivamente a la clase 3. Esto puede lograrse mediante la interpolación de los puntos ya existentes para añadir otros, respetando la geometría existente. La siguiente técnica para paliar este desequilibrio podría aparecer si extraemos una página del libro de ECLAIR y

Pros	Contras
Mayor importancia en el entrenamiento de las clases menos representadas	Entraña riesgo de educar a la red en reconocimiento de patrones erróneos
Permite centrar los esfuerzos en unas pocas clases en vez de todo el conjunto	Puede obligar a la red a necesitar de mayor cantidad de puntos para la identificación, provocando el efecto contrario al deseado
Técnica conocida y de demostrada eficacia en variedad de problemas	Presenta un costo humano en horas de trabajo no trivial

Cuadro 5.3: Pros y contras del data augmentation

utilizamos el pseudo-labelling. Esta técnica consiste en aumentar el número de datos mediante el entrenamiento de un modelo con datos ya etiquetados.

Una vez se ha entrenado, se pasan los datos sin etiquetar por el modelo y la predicción se toma como la clase para esos puntos y se entrena con ambos conjuntos de puntos. Esto ha demostrado ser una gran técnica, aunque no perfecta, para equilibrar las clases (como se vio en el sprint 5).

Pros	Contras
Su eficacia ha sido probada en este mismo proyecto con el dataset de ECLAIR	El modelo ha de estar correctamente entrenado para evitar malas predicciones
Pueden reutilizarse alguno de los modelos entrenados en éste proyecto para pruebas iniciales	En caso de necesitar entrenar un modelo este puede acabar suponiendo los mismos problemas de origen
Genera la mayor cantidad de datos de todas las técnicas	Acarrea un costo de horas de trabajo en el entrenamiento del modelo
No necesita recurrir a alterar la realidad	
Hay puntos de sobra para el entrenamiento en este instante	
Al ser una tecnología semi-supervisada requiere menor costo humano	

Cuadro 5.4: Pros y contras del pseudo-labelling

Por último existe la opción de establecer un proceso de voxelizado de los datos. Esto consiste en realizar una abstracción de la geometría de una escena tridimensional mediante la agrupación de sus puntos en sectores volumétricos llamados vóxeles.

Pros	Contras
Mayor fidelidad que con muestreo aleatorio si el algoritmo está bien diseñado	La existencia de outliers puede sabotear el proceso. El dataset ha de estar muy bien curado
Modificable para dar un peso mayor a las clases menos representadas	Notablemente más costoso computacionalmente que el muestreo aleatorio
No hay riesgo de desaparición de información relevante	Acarrea un costo de horas de trabajo en el entrenamiento del modelo
No requiere supervisión mayor que el resto de métodos	La implementación de CFPS puede resultar compleja

Cuadro 5.5: Pros y contras del voxelizado con CFPS

Esta técnica es ampliamente utilizada en representaciones visuales y abstracciones tridimensionales de datos, pero no por ello está exenta de problemas. La base de esta técnica consiste en encontrar o diseñar un algoritmo de voxelización sólido para una correcta representación de la escena.

Existen técnicas variadas para este propósito. La que aquí se propone es utilizar el algoritmo de Curvature Informed Farhtest Point Sampling [17](CFPS). Este algoritmo es utilizado para encontrar, dentro de cualquier conjunto, los N puntos que garantizan estar a la mayor distancia entre ellos.

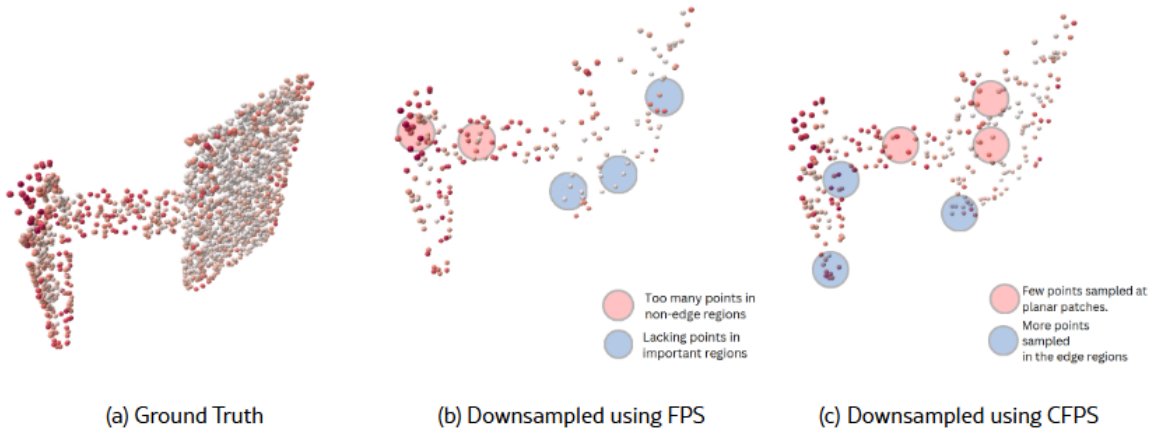

Ilustración 5.2: Comparativa entre FPS y CFPS en geometrías irregulares. Cortesía de la University of Texas at Austin

El algoritmo de CFPS es una mejora sobre el tradicional Farthest Point Sampling (FPS) que aprovecha el concepto matemático de curvatura para encontrar los puntos definitorios de la geometría.

El motivo de uso de este algoritmo es que FPS presenta una gran debilidad a la hora de tratar con datos dispersos. Mientras que FPS funcionaría perfectamente si quisiéramos voxelizar únicamente los edificios, pero debido a la sobrerepresentación de puntos de terreno y lo dispersos que se encuentran los puntos con las clases de cables o torres eléctricas, este método no resulta una opción viable.

5.2.2. Comparativa

Una vez analizadas todas las opciones, es momento de establecer una comparativa entre ellas.

Si observamos los pros y contras establecidos en los cuadros 5.3, 5.4 y 5.5, podemos llegar a varias conclusiones.

La primera de éstas es que se recomienda el uso del pseudo-labelling por encima del data augmentation. En este caso, se tratan ambas como mutuamente exclusivas debido al riesgo del data augmentation de modificar la realidad de la geometría, lo cual puede llevar a un mal aprendizaje del modelo de pseudo-labelling y de los consiguientes.

Los principales motivos para descartar el pseudo-labelling se hallan en el costo de entrenar un modelo de menor escala solo para este proceso y la falta de puntos para entrenar el modelo de primeras. El segundo de los problemas es inexistente en este caso debido a la gran cantidad de puntos con los que cuenta el conjunto de datos de AeroLaser.

El costo de tiempo de entrenar un modelo no puede ser despreciado, por lo que, en caso de no disponer de dicho tiempo, el data augmentation podría ser un recurso viable. En cualquier otro caso se recomienda el uso de pseudo-labelling.

Por último, la modificación del preprocesado para sustituir el muestreo aleatorio por CFPS puede suponer una diferencia crítica. Siempre que la implementación del algoritmo sea sólida y no haya gran presencia de outliers que puedan sabotear el proceso, el voxelizado mediante CFPS debería resultar la mejor de las opciones.

Bibliografía

[1] Aerial Lidar Semantic Segmentation Using PointNet++ Deep Learning.

[2] AeSystem - sistema de sensores para datos geoespaciales.

[3] Arcgis (2024). Clasificación de puntos LiDAR, Documentación.

[4] laspy: Python library for lidar LAS/LAZ IO. — laspy 2.5.0 documentation.

[5] Mathworks (2024). Segmentación semántica.

[6] NumPy user guide — NumPy v2.2 Manual.

[7] Open3D: A Modern Library for 3D Data Processing - Documentación.

[8] PyTorch documentation.

[9] RIEGL VQ-480 II.

[10] What is semi-supervised learning? | IBM.

[11] Wikipedia (2024). K vecinos más próximos.

[12] Wikipedia (2024). LiDAR.

[13] Wikipedia (2024). Multilayer perceptron.

[14] Wikipedia (2024). Voxelización.

[15] Aydin Ayanzadeh. A study review: Semantic segmentation with deep neural networks. *International Journal of Multimedia Information Retrieval*, 03 2018.

[16] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Penksy. Sparse convolutional neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 806–814. IEEE.

[17] Shubham Bhardwaj, Ashwin Vinod, Soumojit Bhattacharya, Aryan Koganti, Aditya Sai Ellendula, and Balakrishna Reddy. Curvature Informed Furthest Point Sampling. version: 1.

[18] Maria Jose Gamez. ONU (2024). Objetivos y metas de desarrollo sostenible.

[19] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys. SEMANTIC3d.NET: A NEW LARGE-SCALE POINT CLOUD CLASSIFICATION BENCHMARK. IV-1/W1:91–98.

[20] Shijie Hao, Yuan Zhou, and Yanrong Guo. A brief survey on semantic segmentation with deep learning. 406:302–321.

[21] M. Kada and D. Kuramin. ALS POINT CLOUD CLASSIFICATION USING POINTNET++ AND KPCONV WITH PRIOR KNOWLEDGE. XLVI-4-W4-2021:91–96. Conference Name: ISPRS TC IV
16th 3D GeoInfo Conference 2021 - 11–14 October 2021, New York City, USA Publisher: Copernicus GmbH.

[22] Sergey Konyakin. Farthest Point Sampling for K-Means Clustering.

[23] Dong-Hyun Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks.

[24] Nan Li, Olaf Kähler, and Norbert Pfeifer. A Comparison of Deep Learning Methods for Airborne Lidar Point Clouds Classification.

[25] Iaroslav Melekhov, Anand Umashankar, Hyeong-Jin Kim, Vladislav Serkov, and Dusty Argyle. ECLAIR: A high-fidelity aerial LiDAR dataset for semantic segmentation.

[26] A. Nurunnabi, F. N. Teferle, J. Li, R. C. Lindenbergh, and S. Parvaz. INVESTIGATION OF POINTNET FOR SEMANTIC SEGMENTATION OF LARGE-SCALE OUTDOOR POINT CLOUDS. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVI-4/W5-2021:397–404, 2021.

[27] Yassine Ouali, Céline Hudelot, and Myriam Tami. An Overview of Deep Semi-Supervised Learning.

[28] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation.

[29] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.

[30] Nina M. Singer and Vijayan K. Asari. DALES Objects: A Large Scale Benchmark Dataset for Instance Segmentation in Aerial Lidar. 9:97495–97504. Conference Name: IEEE Access.

[31] M. Soilán, A. Nóvoa, A. Sánchez-Rodríguez, B. Riveiro, and P. Arias. SEMANTIC SEGMENTATION OF POINT CLOUDS WITH POINTNET AND KPCONV ARCHITECTURES APPLIED TO RAILWAY TUNNELS. V-2-2020:281–288.

[32] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6410–6419. IEEE.

[33] Sik-Ho Tsang. Review — Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks.

[34] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data.

[35] Nina Varney, Vijayan K. Asari, and Quinn Graehling. DALES: A Large-scale Aerial LiDAR Data Set for Semantic Segmentation.

[36] Xu Yan. Pointnet/pointnet++ pytorch. *https://github.com/yanx27/Pointnet*$_P$*ointnet*2$_p$*ytorch*, 2019.