CEIG – Spanish Computer Graphics Conference (2019)
A. Jarabo and D. Casas (Editors)

A voxel-based deep learning approach for Point Cloud Semantic Segmentation

Miguel Díaz-Medina[1]*, José-Manuel Fuertes-García[1], Carlos-Javier Ogayar-Anguita[1], Manuel Lucena[1]

[1]Departamento de Informática. Universidad de Jaén

Abstract

Semantic segmentation has been a research topic in computer vision for decades. This task has become a crucial challenge nowadays due to emergence of new technologies such as autonomous driving. Nonetheless, most existing segmentation methods are not designed for handling the unstructured and irregular nature of 3D point clouds. We propose a voxel-based technique for 3D point cloud data semantic segmentation using 3D convolutional neural networks. It uses local voxelizations for learning spatial patterns, and also corrects the imbalance of the data, something very problematic with 3D datasets.

CCS Concepts

•***Computing methodologies*** → *Computer Graphics; Machine Learning;*

1. Introducción

La segmentación de nubes de puntos es un área de gran interés en la que la mayoría de los métodos están todavía evolucionando. Existen numerosos ámbitos donde los avances en esta línea son imprescindibles, como la gestión inteligente del territorio, la planificación urbana, los vehículos autónomos, el modelado BIM o la arqueología, entre otros. En estos casos los datasets son de una gran magnitud, y los métodos clásicos, puramente geométricos, no trabajan correctamente o bien necesitan un tedioso procesamiento manual adicional. Por esta razón, es necesario automatizar el proceso en la medida de lo posible.

El objetivo de nuestra propuesta es un sistema de segmentación de nubes de puntos que las utiliza como entrada en su formato irregular y segmenta semánticamente los datos a partir de patrones aprendidos. Proponemos una técnica de regularización de la nube de puntos, que genera grids de vóxeles en diferentes entornos locales de la nube, espacio donde se asignará una etiqueta de clase. En base a este conjunto de datos voxelizados, se propone una arquitectura de red neuronal convolucional que contiene capas de convolución 3D que procesan los grids de entrada, extrayendo patrones en forma de características. De esta forma la red infiere la etiqueta de cada punto a partir de la morfología de su entorno local.

2. Trabajos previos

Cuando se decide aplicar aprendizaje profundo sobre modelos tridimensionales, existen varias alternativas para la representación de los datos: nubes de puntos, voxelizaciones, mallas de triángulos e imágenes RGB(D). Una red neuronal convolucional siempre debe usar datos de entrada del mismo tamaño. Al tratar imágenes, la resolución deberá ser fija (p. ej.: 1024×1024). En el caso de datos 3D, las representaciones basadas en imágenes o en vóxeles son las más adaptables, pero sufren de baja precisión y un uso excesivo de memoria en el caso de las voxelizaciones.

Existen métodos que proponen diseños de redes que aceptan conjuntos de puntos llevando a cabo un muestreo sobre la nube original, y trabajando siempre con el mismo número de elementos, lo que implica una pérdida importante de precisión. Destaca en este sentido PointNet++ [QYSG17], que está más enfocado a la clasificación de subconjuntos de puntos segmentados de la nube original. Por otra parte, existen otros métodos de clasificación que proponen la voxelización completa de objetos para entrenar una red neuronal convolucional 3D [MS15] [QSN*16]. Este enfoque también requiere modelos segmentados, a los que asignará una única etiqueta.

La mayoría de las propuestas actuales que tratan nubes de puntos mediante aprendizaje automático procesan modelos segmentados para poder realizar una clasificación. Este trabajo propone un algoritmo de segmentación semántica basado en técnicas de etiquetado de nubes de puntos [HY16] que permita realizar la tarea minimizando el conocimiento previo requerido.

3. Sistema propuesto

Nuestro enfoque se basa en la voxelización de entornos locales de una nube de puntos para entrenar una red neuronal que aprende patrones espaciales. Para ello, deben utilizarse muestras de tamaño

* Autor para correspondencia. E-mail: mdm00030@red.ujaen.es

DOI: 10.2312/ceig.20191206

fijo, ya que esta es una condición de las redes neuronales convolucionales para los datos de entrada [GBC16], lo que supone que siempre se consumirán grids de las mismas dimensiones. Hay que destacar que este enfoque no se basa en una voxelización global en un sentido estricto, esto es, la construcción de un solo modelo de vóxeles para toda la nube, sino en la generación de un número determinado de grids de vóxeles de tamaño fijo en entornos locales a lo largo de dicha nube que se corresponderán con ejemplos de entrenamiento del dataset procesado. De esta forma nuestro sistema utiliza una red neuronal convolucional 3D que segmenta semánticamente nubes de puntos tomando como entrada un conjunto de grids de vóxeles de iguales dimensiones. Los datos de partida para el entrenamiento de la red lo constituyen nubes de puntos previamente segmentadas, donde cada punto ya tiene asignada una etiqueta. Este método de aprendizaje automático no necesita de edición manual ni de algoritmos basados en geometría, normalmente menos eficientes y precisos. Sin embargo, en la actualidad, son precisamente éstos métodos los que permiten tener conjuntos de datos segmentados para el entrenamiento de las redes neuronales. Además, éstas necesitan una gran cantidad de datos de entrenamiento para ser efectivas.

3.1. Preprocesamiento de datos

Las nubes de puntos utilizadas en ámbitos como la arquitectura, la planificación urbana, la gestión del territorio o la arqueología, constituyen conjuntos de datos muy densos, especialmente cuando proceden de escaneados LiDAR. Para poder tratarlos eficientemente con este enfoque de red neuronal, es necesario realizar un proceso de muestreo y ajuste sobre los datos espaciales de entrada, ya que debe buscarse un compromiso entre la maximización del volumen espacial a estudiar y la minimización de la cantidad de puntos a tratar. Además, todo algoritmo de aprendizaje necesita de un conjunto de datos balanceado (equilibrado) para asegurar un entrenamiento correcto, lo que supone disponer de una proporción similar en el número de ocurrencias (puntos) de cada etiqueta semántica (clase).

3.1.1. Muestreo

Cuando se procesan nubes de puntos muy densas, esto es, de varios cientos de millones de puntos, surgen serios problemas de rendimiento en el manejo de esta información con redes neuronales. Por esta razón es necesario seleccionar un subconjunto representativo de los puntos de entrada. Para realizar este muestreo se establece un nivel de precisión, P, que será la distancia media entre cualesquiera dos puntos de la nube muestreada. La mínima caja envolvente de la nube de entrada se divide en un grid o espacio discreto de $I \times J \times K$ celdas. Cada celda de dicho espacio encierra un conjunto arbitrario de puntos, de entre los cuales se selecciona aleatoriamente un representante (para evitar un sesgo del método), que pasará a formar parte de la nueva nube muestreada. La elección de la precisión será crucial, pues se debe mantener en todo momento la información más relevante de la nube. Si P es demasiado grande, se perderán los detalles y el método no dará buenos resultados.

3.1.2. Balanceo del dataset

La mayoría de los datasets disponibles para aprendizaje automático no están balanceados, lo que supone que la proporción de elementos de cada clase es desigual. En el caso de las nubes de puntos, este problema se acentúa. Por ejemplo, la mayoría de puntos de una nube LiDAR pertenecen a la clase *terreno* en escenas rurales, y a la clase *edificio* en escenas urbanas. Sin embargo, hay que seguir tratando correctamente otras clases minoritarias como *coche* o *tendido eléctrico*.

La solución consiste en muestrear aleatoriamente N puntos de cada clase disponible en la nube [HY16], siendo N es el número de puntos de la clase con menos ocurrencias. Como el muestreo es aleatorio sin reemplazamiento, el espacio quedará cubierto prácticamente en su totalidad. Es decir, habrá puntos de cualquier región del espacio, y no solamente de una zona aislada de la nube. Posteriormente, estos puntos seleccionados serán los centros de cada una de las voxelizaciones locales con las que se alimentará a la red neuronal, si bien en dichas voxelizaciones no solo se tendrán en cuenta los puntos seleccionados como representantes de clase, sino todos los puntos originales de la nube para no perder información.

3.1.3. Aumento del dataset

Uno de los problemas que presentan las redes neuronales en general, y las convolucionales en particular, es que necesitan un gran volumen de datos para conseguir extraer patrones. A diferencia de los humanos, necesitan procesar el mismo objeto desde diferentes ángulos y puntos de vista para poder reconocerlo posteriormente.

Para compensar este hecho, se aplican 12 transformaciones sucesivas de rotación sobre cada entorno local al hacer la selección de puntos [MS15]. Es decir, después de seleccionar el punto central para cada voxelización local, y tomar los puntos contenidos en el correspondiente grid, se aplican 12 rotaciones acumuladas de 30 grados cada una alrededor del eje Y en el sistema de coordenadas de la mano derecha. Esto enseñará a la red diferentes puntos de vista de cada zona espacial, potenciando el aprendizaje.

3.2. Voxelizaciones locales

El núcleo del método propuesto se basa en la generación de voxelizaciones locales de tamaño fijo. Dada una nube de puntos definida sobre el espacio métrico, donde cada punto tiene coordenadas (x,y,z), se seleccionan N puntos con los criterios presentados en 3.1.1, y se realiza una voxelización local centrada en cada uno de dichos puntos. Cabe destacar que no es posible realizar una voxelización global sobre toda la nube para posteriormente seleccionar subregiones y ganar en rendimiento, dado que las regiones alrededor de cada punto seleccionado no estarán perfectamente alineadas. Para conseguir esto, la distancia entre puntos vecinos debería ser uniforme en cada una de las tres dimensiones para toda la nube.

Para una correcta identificación de patrones espaciales, cada punto analizado debe estar acompañado de la información de sus vecinos más cercanos. Es necesario definir el tamaño de las voxelizaciones locales, $N \times N \times N$, y también la dimensión de cada vóxel, t, que determina la porción del espacio métrico que ocupa. Así, para un punto (x,y,z), el grid generado tendrá una caja envolvente de $[x-\frac{N\times t}{2}, x+\frac{N\times t}{2}] \times [y-\frac{N\times t}{2}, y+\frac{N\times t}{2}] \times [z-\frac{N\times t}{2}, z+\frac{N\times t}{2}]$. La voxelización será diferente en las fases de entrenamiento y de test. En la Figura 1 vemos un ejemplo de la voxelización local.

En el entrenamiento se dispone de cada punto etiquetado con la

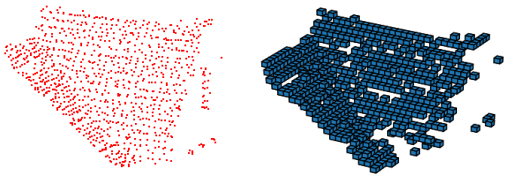

Figure 1: *Voxelización de un entorno local de* 28^3.

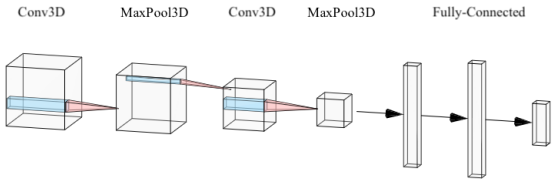

Figure 2: *Red Neuronal Convolucional 3D propuesta.*

clase a la que pertenece (aprendizaje supervisado, [GBC16]). Para cada punto seleccionado se genera un grid de vóxeles tomando dicho punto como centro. Posteriormente se toman los puntos vecinos que encierra dicho grid y se etiquetan los vóxeles que contienen algún punto con el valor de clase del punto central a partir del cuál se generó la voxelización local. En este trabajo consideraremos únicamente la ocupación para el etiquetado de vóxeles.

Durante el test, es proceso de voxelización es básicamente el mismo, con la diferencia de que el grid de vóxeles no se etiqueta con ninguna clase. El objetivo será saber qué etiqueta le corresponde al grid con el que se alimenta la red. Como normalmente en un vóxel habrá más de un punto, la alternativa de generar la voxelización local para cada uno de los puntos no es eficiente, ya que los cálculos serían muy redundantes. Así, dada una nube, se calcula su mínima caja envolvente completa, y se divide en regiones de tamaño t (la anchura del vóxel). Posteriormente se escogen los puntos centrales de las subdivisiones, y se genera una nueva nube, resultado de muestrear la primera. A continuación, se generan grids locales para todos los puntos de la nueva nube, se clasifican, y entonces se le asigna a cada punto de la nube original la etiqueta del punto de la nueva nube que está a menor distancia de él.

3.3. Arquitectura de la red

Nuestra propuesta de arquitectura (figura 2) parte de dos capas de *convolución 3D*, seguida cada una de una capa de *max pooling 3D*, concluyendo con una *red fully-connected* para clasificar entre las distintas clases con las que haya sido entrenada. Este trabajo se inspira en las arquitecturas habituales para la clasificación de imágenes, extendidas en este caso a 3D. La convolución 3D [SLJ*15] surgió inicialmente para extraer patrones de datos de vídeo, aunque en nuestro caso se aplica para extraer patrones en forma de voxelización o grid discreto, con los que se alimentará la red. Las capas de *pooling* [GBC16] sirven para reducir la dimensión de los datos y resumir características, y se aplican tras cada capa de convolución. Con este paradigma se presenta el siguiente ejemplo: 20 filtros en las dos primeras capas de convolución, un grid de entrada de 20^3 vóxeles, un tamaño de kernel para las convoluciones de 5^3 con un *stride* de 1 (desplazamiento del kernel en cada convolución), un tamaño de kernel de 2^3 con un *stride* de 2 en *max pooling*; y finalmente una capa *fully-connected* con una dimensión de grid de 2 x 2 x 2 y 20 filtros (lo que resulta en un vector de características unidimensional de tamaño 160). La parte *fully-connected* de la red está compuesta por 3 capas: de entrada, oculta y de salida, con 160, 300, y M unidades respectivamente con la configuración anterior. M es el número de clases a distinguir en el problema.

4. Experimentos

4.1. Dataset

Todo algoritmo de aprendizaje automático necesita datos para ajustar un modelo mediante entrenamiento y poder validarlo posteriormente. En el caso del aprendizaje profundo supervisado debe conocerse además la clase a la que pertenece cada dato [GBC16]. Esto resulta muy complicado al trabajar con nubes de puntos, ya que existen muy pocos datasets previamente clasificados para realizar los entrenamientos. En este trabajo se ha utilizado Semantic 3D [HSL*17], un framework para la evaluación de algoritmos de segmentación/clasificación de nubes de puntos que contiene en torno a 4.000 millones de puntos etiquetados manualmente con dimensiones no normalizadas. El dataset de prueba está compuesto por un total de 15 nubes de puntos de escenas rurales, urbanas y suburbanas, etiquetadas convenientemente para el entrenamiento con 8 clases: 1-terreno artificial, 2-terreno natural, 3-vegetación alta, 4-vegetación baja, 5-edificios, 6-paisaje accidentado, 7-artefactos de escaneo y 8-coches. En este trabajo se ha adaptado el dataset mediante las modificaciones que se describen a continuación.

4.2. Muestreo y balanceo

Tras una serie de pruebas y teniendo en cuenta la escala de los objetos presentes en el dataset, se ha decidido utilizar un tamaño de vóxel de 0.2 metros. Con nubes muy densas este hecho puede utilizarse para reducir el volumen de datos mediante un muestreo. Si dicho muestreo se realiza con distancias entre puntos menores que el tamaño de vóxel se produce oversampling y se desperdicia memoria. Usar distancias mayores del tamaño de vóxel provocará una pérdida de información severa al romper la continuidad entre muestras (undersampling). Por tanto, para garantizar el mejor aprovechamiento de la nube original sin pérdida de información, se utiliza una resolución de muestreo igual al tamaño de vóxel, consiguiendo reducir el dataset de entrada de 280 millones de puntos a unos 0,43 millones de puntos sin perder información relevante. Debe recordarse que este muestreo de la nube original sirve para construir las voxelizaciones locales para el entrenamiento de la red. Durante el test, todo el dataset original será clasificado.

Además de lo anterior, el dataset no está balanceado, como es habitual en escaneados de escenas exteriores. Aleatoriamente se seleccionan de cada categoría tantos puntos como contenga la clase minoritaria. Posteriormente, para cada punto seleccionado se realiza una voxelización local utilizando rotaciones incrementales de 30 grados hasta completar una vuelta. Para poder entrenar el mode-

lo y validarlo, el dataset se divide en una proporción de 80 % para entrenamiento y 20 % para test (inferencia).

4.3. Parámetros e hiperparámetros

Las parámetros de la red se ajustan tras una serie de experimentaciones donde se comprueba la convergencia del error de clasificación cometido. Con las nubes de puntos de prueba, el mejor valor para la dimensión de las voxelizaciones locales ha sido de 28^3, siendo el tamaño de vóxel de 0,2 metros.

En cuanto a la arquitectura de la red, las capas de convolución aprenden un total de 20 filtros cada una, con un tamaño de núcleo de 5^3 y un *stride* de 1. Las capas de *pooling* tienen un tamaño de núcleo de 2^3 y un *stride* de 2. Con esta configuración, la primera capa de la red *fully-connected* ha de tener 1280 unidades, seguida por una capa de *dropout*. La segunda capa está fijada a 300 unidades ocultas, seguidas por otra capa de *dropout*. La última capa tiene 8 unidades, una por cada clase del problema. Las capas de *dropout* se emplean para evitar el sobreaprendizaje de la red [GBC16].

Como hiperparámetros, se establece un entrenamiento de 60 *epochs* (iteraciones sobre todos los datos del conjunto de entrenamiento), con un *ratio de aprendizaje* de 10^{-4} y una *disminución del ratio de aprendizaje* del 95 % en cada *epoch* [Ama93] [Sch15]. El tamaño de *batch* [GBC16] estará fijado a 32. Como algoritmo de optimización se utiliza el *Gradiente Descendiente Estocástico* con $Momentum = 0{,}9$ y como función de coste se utiliza *Entropía Cruzada* [GBC16].

4.4. Resultados obtenidos

Con la configuración anterior y una división entrenamiento/test del 80/20 %, después de 60 epochs se consigue una precisión en la segmentación de un 86,601 % en el entrenamiento y un 86,376 % en el test. El hecho de estar tan igualados indica que el entrenamiento se ha realizado correctamente y el modelo no se ha sobreajustado a los datos de entrenamiento. En la figura 3 se observa el resultado de la segmentación.

4.5. Rendimiento

Los experimentos se han realizado sobre una máquina con un procesador Intel Core i7 8700K, 64 GB de memoria RAM y una GPU NVIDIA Titan Xp. El proceso de generación del conjunto de entrenamiento y test se completa en un total de 2 horas, mientras que el entrenamiento de la red tarda en torno a 8 horas utilizando aceleración por GPU mediante CUDA usando el framework Torch.

5. Conclusiones

En este trabajo se presenta un método de segmentación semántica de nubes de puntos basado en la utilización de redes neuronales convolucionales 3D que trabajan con voxelizaciones locales para identificar patrones espaciales. El sistema permite trabajar con una resolución determinada realizando un muestreo sobre los datos originales para reducir el volumen de los datos de entrada, sin que ello afecte a la capacidad de aprendizaje de la red. También soluciona el problema de los datos no balanceados, algo muy problemático en los escaneados de exteriores, donde este factor es extremo.

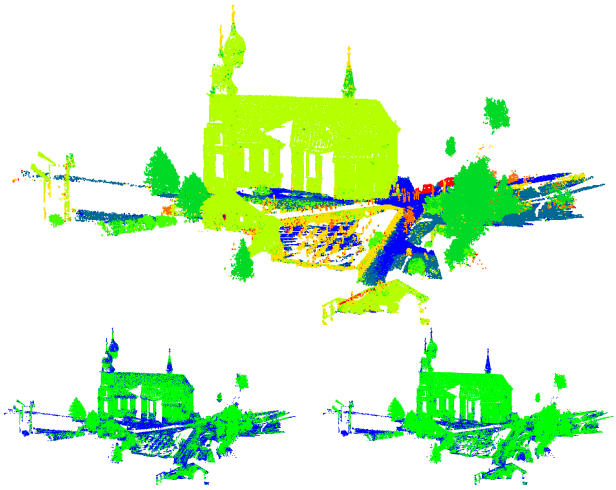

Figure 3: *Arriba: etiquetado durante la validación tras 60 epochs de entrenamiento. Abajo: se muestra cómo la red consigue mejorar desde el epoch 1 (izquierda) al epoch 60 (derecha). En verde los puntos bien clasificados, en azul los mal clasificados.*

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia, Innovación y Universidades y la Unión Europea a través del proyecto RTI2018-099638-B-I00.

References

[Ama93] AMARI S.-I.: Backpropagation and stochastic gradient descent method. *Neurocomputing 5* (06 1993), 185–196. 4

[GBC16] GOODFELLOW I., BENGIO Y., COURVILLE A.: *Deep Learning*. MIT Press, 2016. 2, 3, 4

[HSL*17] HACKEL T., SAVINOV N., LADICKY L., WEGNER J. D., SCHINDLER K., POLLEFEYS M.: SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2017), vol. IV-1-W1, pp. 91–98. 3

[HY16] HUANG J., YOU S.: Point cloud labeling using 3d convolutional neural network. 1, 2

[MS15] MATURANA D., SCHERER S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2015), p. 922 – 928. 1, 2

[QSN*16] QI C. R., SU H., NIESSNER M., DAI A., YAN M., GUIBAS L. J.: Volumetric and multi-view cnns for object classification on 3d data. *CoRR abs/1604.03265* (2016). 1

[QYSG17] QI C. R., YI L., SU H., GUIBAS L. J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems 30*. 2017, pp. 5099–5108. 1

[Sch15] SCHMIDHUBER J.: Deep learning in neural networks: An overview. *Neural Networks 61* (2015), 85–117. 4

[SLJ*15] SZEGEDY C., LIU W., JIA Y., SERMANET P., REED S., ANGUELOV D., ERHAN D., VANHOUCKE V., RABINOVICH A.: Going deeper with convolutions. In *CVPR* (2015). 3