Estudio y aplicación de las redes neuronales convolucionales 3D

Trabajo de final de grado realizado en la
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
por

Mario Gimenez Arnal

En cumplimiento parcial
de los requerimientos para el grado en
Ingenieria de Sistemas Audiovisuales

Tutor: Javier Ruiz Hidalgo

Barcelona, Octubre 2018

Resumen

El objetivo de este proyecto es el estudio de las redes neuronales, su entrenamiento y aplicación junto con la creación de una base de datos adecuada al sistema.
En concreto se estudiarán las redes convolucionales 3D aplicadas a una base de datos generada a partir de nubes de puntos tridimensionales, a la que se le aplica un proceso de voxelización.
La base de datos se genera a partir de parejas de nubes de puntos que representan una misma escena vista desde puntos de vistas distintos de las cuales se extrae un punto clave de cada nube, que denominaremos keypoint, y los puntos de su alrededor, con el objetivo de utilizarlos para entrenar la red neuronal convolucional que se encargará de decidir si cada pareja de keypoints corresponden al mismo punto de la escena, o no.

Resum

L'objectiu d'aquest projecte es l'estudi de les xarxes neuronals, el seu entrenament i aplicació juntament amb la creació d'una base de dades adient pel sistema.
En concret s'estudiaran les xarxes convolucionals 3D aplicades a una base de dades generada a partir de núvols de punts tridimensionals, a la que se li aplica un procés de voxelització.
La base de dades es genera a partir de parelles de núvols de punts que representen una mateixa escena vista des de punts de vista diferents, d'aquests núvols, s'extrau un punt clau de cada núvol, que denominarem keypoint, i punts del seu voltant, amb l'objectiu d'utilitzar-los per entrenar la xarxa neuronal convolucional que s'encarregarà de decidir si cada parella de keypoints corresponen al mateix punt de l'escena, o no.

Abstract

The purpose of this project is the study of neural networks, their training and application together with the creation of a database appropriate to the system.
Specifically, the 3D convolutional networks applied to a database generated from three-dimensional point clouds, to which a voxelization process is applied, will be studied.
The database is generated from pairs of point clouds that represent the same scene seen from different points of view from which a key point is extracted from each cloud, which we will call keypoint, and the points around it, with the aim of using them to train the convolutional neural network that will decide whether each pair of keypoints correspond to the same point of the scene, or not.

Agradecimientos

Primerament vull agrair al Javier Ruiz Hidalgo, per haver-me tutoritzat el projecte, per haver-me guiat durant aquest procés i per la paciència que ha tingut durant aquest temps
També donar-li les gràcies a l'Albert Gil, Alba Pujol i Miguel Humberto per l'ajuda que m'han donat durant aquest camí.
Al meu pare i a la meva mare, dir-los que ho han aconseguit, independentment d'on siguin
Als meus germans i germanes, perquè ells m'han ensenyat a viure.
I finalment, a totes les persones amb les quals he compartit aquest camí, ho heu fet únic!

Y ahora, prefiero ser un indio, que un importante abogado...

Contents

1 Introducción **7**
1.1 Motivación . . . 7
1.2 Objetivos . . . 7
1.3 Plan de trabajo . . . 8
1.4 Incidencias . . . 9

2 Conceptos Básicos **11**
2.1 Redes Neuronales . . . 11
2.2 Redes neuronales convolucionales . . . 12
2.3 Función activación . . . 13
2.4 Dropout . . . 14
2.5 Procedimiento . . . 15

3 Estado del arte **17**
3.1 Trabajos relacionados . . . 17
3.1.1 SEGCloud:Semantic Segmentation of 3DPoint Clouds [1] . . . 17
3.1.2 Real-Time Plane Segmentation and Obstacle Detection of 3D Point Clouds for Indoor Scenes[2] . . . 17
3.1.3 Voxel and graph-basede point cloud segmentation of 3D scenes using perceptual grouping laws[3] . . . 18

4 Metodología **19**
4.1 Voxelización . . . 19
4.2 Generación de la base de datos . . . 22
4.3 Diseño y entrenamiento de la red neuronal . . . 24
4.4 Herramientas implementación . . . 26
4.4.1 TensorFlow . . . 27
4.4.2 Torch . . . 27
4.4.3 Keras . . . 27

5 Resultados **28**

6 Presupuesto **34**

7 Conclusiones **35**

8 Bibliografia **36**

List of Figures

1 *Diagrama de Gantt parte 1* 8
2 *Diagrama de Gantt parte 2* 9
3 *Capa oculta* 12
4 *Expresión de la funcion softmax* 13
5 *Función Sigmoid* 13
6 *Función ReLu* 14
7 *Bloques que forman el proyecto* 19
8 *Nube de puntos* 21
9 *Voxelización 10x10x10* 22
10 *Ocupación en cada voxel* 22
11 *Estructura de una rama* 25
12 *Estructura fully connected* 26
13 *Estructura completa* 28
14 *1a Prueba* 29
15 *Test sobreentrenamiento* 30
16 *Test curba learning rate* 31
17 *Estructura 8x8x8* 32
18 *Estructura 4x4x4* 33

1 Introducción

1.1 Motivación

Desde un inicio tenia claro que eltema de mi trabajo de final de grado estaría relacionado con la inteligencia artificial, ya que a parte que es una temática que me genera un gran interés, es un campo que esta en continua evolució y que cada vez lo va a estar mas gracias a la mejora de la tecnologia.
Al plantearle esta opción a mi tutor , y después de tener en cuenta otras opciónes, decidimos realizar una aplicación de las redes neuronales, en la que se encuentran correspondencias entre dos puntos de dos nubes de puntos diferentes, que representan una misma escena.
En los últimos años ha aumentado la utilización y campo de aplicación de los sensores 3D. Particularmente ha crecido enormemente en el ámbito de la realidad virtual y el entrelazado entre una escena real con algún algoritmo, objeto, etc. virtual. Para que esto sea posible, es necesaria la identificación y relación entre los puntos de la realidad, con los que son generados virtualmente. En este proyecto, parte de este trabajo de identificación, es realizado mediante una red neuronal.

1.2 Objetivos

El principal objetivo de este proyecto es desarrollar un sistema, haciendo uso de las redes convolucionales 3D, que pueda llevar a cabo una clasificación entre parejas de puntos cada uno de ellos extraídos de una misma escena, vista desde dos puntos distintos.

Para ello será importante el hacer un estudio sobre la aplicación de las redes neuronales en estos campos realizados en otros artículos o proyectos.
Otro objetivo importante es definir y optimizar la manipulación y tratado de los datos de entrada a la red neuronal, en este caso nubes de puntos 3D,es decir, un conjunto de puntos situados en una coordenadas de 3 dimensiones, de manera que se obtengan los mejores resultados disminuyendo el tiempo utilizado

1.3 Plan de trabajo

El proyecto ha sido dividido en tres partes, la primera dedicada a la búsqueda de información de las diferentes partes del proyecto, la segunda dedicada a la implementación del codigo necesario para generar la base de datos y el funcionamiento de la red neuronal, y la última parte de recopilación y análisis de resultados.
Junto a estas tres partes, hay una parte previa de planteamiento de proyecto. En la siguiente figura se presenta el diagrama de Gantt indicando las tareas que componen cada una de estas partes mayores.

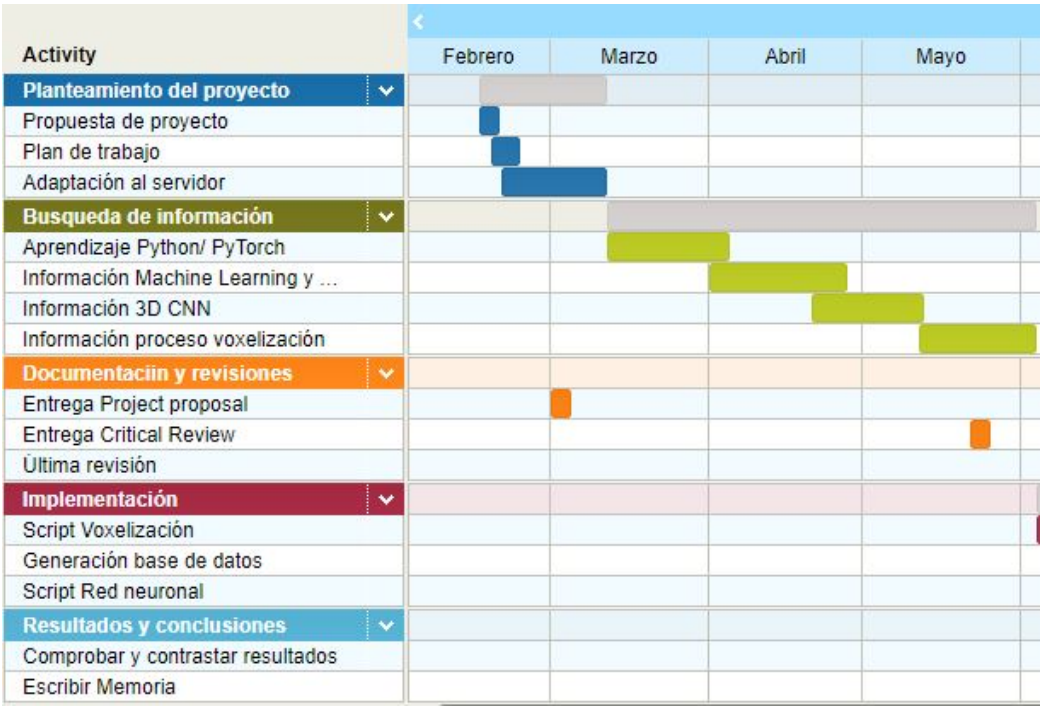

Figure 1: *Diagrama de Gantt parte 1*

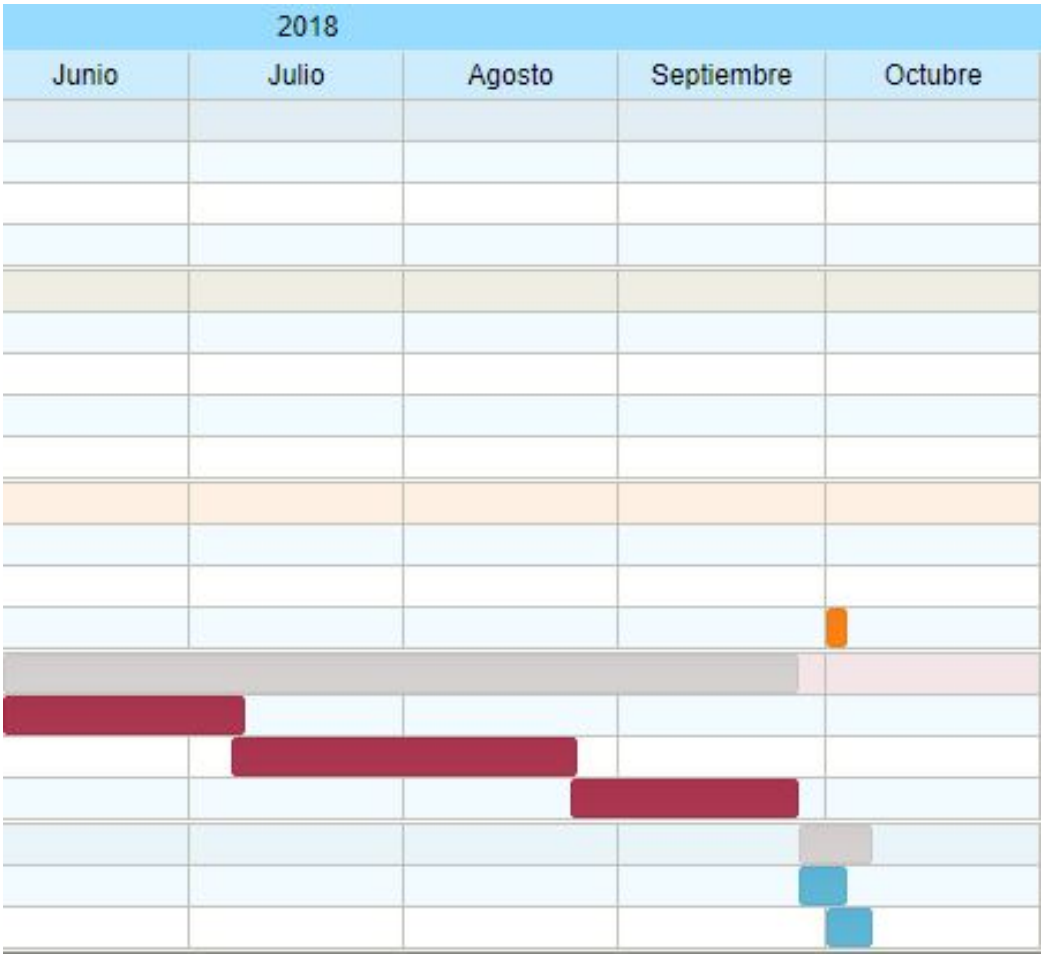

Figure 2: *Diagrama de Gantt parte 2*

1.4 Incidencias

En este apartado se explicaran las incidencias que se han dado durante el transcurso del proyecto.
La primera pequeña incidencia del proyecto, fué la adaptación al sistema de trabajo. Poco acostumbrado a trabajar en linux y novato utilizando python, la adaptacion al trabajo en el servidor de la universidad se alargó mas de lo esperado.
La siguiente incidencia, que en este caso si que generó un aumento del tiempo considerable, fue la errónea generación de la base de datos que iba a utilizar para entrenar, validar y testear el sistema. Fue a causa de un error de código, y al ver junto a mi tutor que el sistema no funcionaba, revisamos todo lo realizado hasta entonces para descubrir donde estaba el fallo, hallando que el

problema estaba en la base de datos. Así que tuve que generar de nuevo toda la base de datos para poder seguir con el proyecto.

Finalmente, la parte del trabajo que mas tiempo a consumido, han sido los procesos computacionales, es decir: la realización de los tests, y sobretodo la generación de la base de datos. En el caso de los tests, en parte fue error mio ya que al principio no utilizaba la GPU de la que disponía en el servidor de la UPC. Al principio el proceso podia tardar entre unas 30 o 40h, pero al empezar a utilizar la GPU el tiempo de este proceso fue reducido a unas 4h. A parte el tiempo de generación de la base de datos, en el que lanzaba mas de 20 procesos en rondas de 8 procesos paralelos en los que algunos de estos procesos podían tardar también unas 30h.

2 Conceptos Básicos

En este apartado se plantearan los conceptos básicos necesarios para diseñar un sistema relacionado con las redes neuronales convolucionales, empezando desde el concepto de red neuronal, pasando por las diferentes partes que conforman una red neuronal y finalizando con el proceso estándar para trabajar con dicho sistema.

2.1 Redes Neuronales

El ámbito del *deep learning* se basa en la utilización de las denominadas redes neuronales, que son un sistema automático de aprendizaje inspirado en el sistema nervioso biológico. Este sistema sigue una estructura de capas cada una de las cuales, esta formada por un conjunto de neuronas, que seria la estructura más básica de este sistema. Estas neuronas, están interconectadas con neuronas de otras capas mediante un enlace, el cual esta ponderado por unos pesos que multiplican la información que lo atraviesa,y que a la hora de entrenar el sistema, se va modificando adaptándose al resultado deseado. Estrutcturalmente, esta estructura se divide en tres tipos de capas:

- Capas de entrada:
  Es la conjunto de neuronas mediante el que la información accede a la red. No se realiza ningun procesado en esta capa

- Capas ocultas:
  Son las capas que reciben la información de las capas de entrada. A la información que les llega, son sumadas mediante una función de transferencia en al que se le puede aplicar un peso diferente a la información que llega de cada neurona. A esta unión se le aplica una función de activación. Esta función es una de las partes más importantes de la red. Es obligatorio que esta función sea no lineal, ya que de lo contrario al concatenar capas con funciones lineales, se podría resumir finalmente en una sola capa lineal y de esta manera no se podrían obtener buenos resultados en el proceso.

- Capa de salida:
  Es la capa encargada de recopilar la información y proveer una salida dependiendo del tipo de red y la clasificación que se quiera realizar.

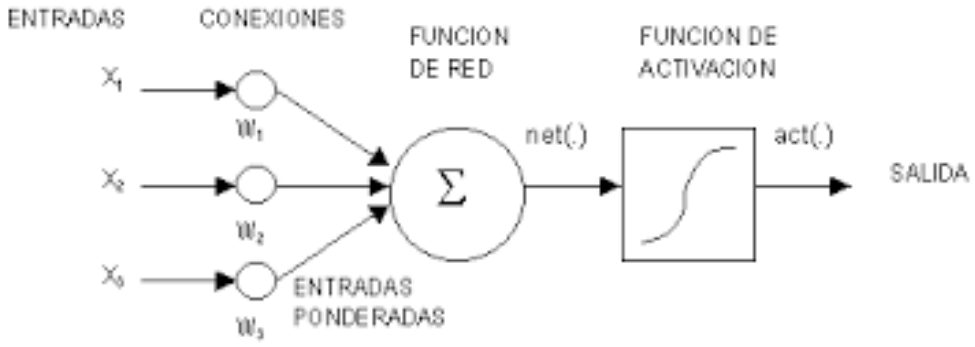

Figure 3: *Capa oculta*

2.2 Redes neuronales convolucionales

Un tipo de redes neuronales, que es el utilizado en este proyecto, son las redes neuronales convolucionales.

En estas capas, en lugar de tener pesos independientes por cada neurona, los pesos se utilizan para definir un filtro convolucional que se aplica a la información de entrada para a continuación, aplicar la función de activación al resultado de la convolución. En una capa convolucional, hay varios parámetros a definir: El primero es el tamaño del filtro convolucional,seguido del paso de convolución (o stride) que define el desplazamiento que aplicamos al filtro a la hora de realizar la convolución, y finalmente definir como operaremos con los bordes, si rellenaremos la parte exterior de la imagen de entrada para que el filtro pueda operar en esa zona, o si elegiremos que el filtro tan solo opere dentro de la zona de la imagen (se denomina padding a este proceso). Según lo que escojamos en relación a estos parámetros, definiremos la dimensión que tendrá la información a la salida de la capa.
Ya en la parte final de la red neuronal convolucional, nos solemos encontrar una capa fully connected, seguida de una una capa que transforma los valores a probabilidad.
Las capas fully connected, tiene la caracteristica que todas las neuronas están interconectadas con las neuronas de la capa anterior y posterior. Esta característica no las hace especialmente útiles cuando se trabaja con una gran cantidad de nodos, ya que las conexiones aumentan con mucha rapidez.
Finalmente, hay que aplicar una última capa, que transforme los valores de salida de la capa fully connected en valores probabilísticos, de manera que tengamos un valor adecuado para poder realizar la clasificación.
Ejemplos de estas capas podrían ser la capa softmax, que recibiendo la misma

cantidad de neuronas que clases, asigna una probabilidad a cada una de las clases en función de los valores recibidos y que definida por la siguiente función mostrada a continuación, o una capa sigmoid, muy utilizada porque sus valores de salida van de 0 a 1, coincidiendo con los valores posibles de la probabilidad.

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{j} e^{y_j}}$$

Figure 4: *Expresión de la funcion softmax*

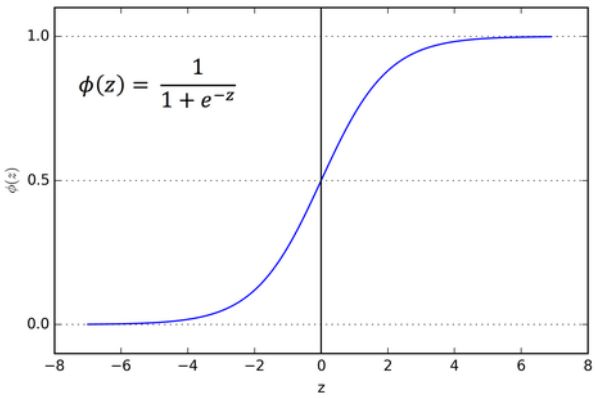

Figure 5: *Función Sigmoid*

2.3 Función activación

Como se ha comentado anteriormente, la función de activación, se aplica justo a la salida de una capa de la red, con el objetivo de aplicar una no linealidad a la información de salida. Al aplicar estas funciones, aparte de evitar el hecho de que se pueda representar toda la red neuronal con una sola función lineal, conseguimos aumentar la capacidad de la red de aproximarse a muchos mas tipos de funciones y aumenta su eficacia a la hora de clasificar. Una de las funciones de activación mas utilizadas en la actualidad, y que será la utilizada en este proyecto es la ReLU (Rectified Linear Units) que estan definidas por la siguiente función:

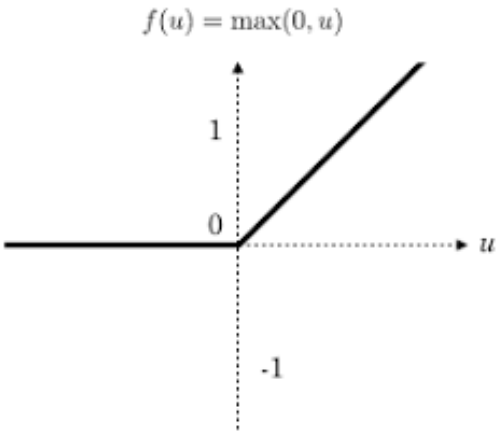

Figure 6: *Función ReLu*

Estas funciones, tienen la ventaja de que aceleran el proceso de entrenamiento, y son muy fáciles de implementar, razones por las que son tan utilizadas. Pero también es cierto que presentan alguna desventaja, como el hecho de que algunos pesos, puedan tender a estancarse en cero, y no ser nunca activados.
Otra función de activación utilizada es la funcion sigmoide. Esta función es muy utilizada dado que sus valores de salida, se en

2.4 Dropout

Se refiere al proceso de no considerar los valores de ciertas neuronas aleatorias. Al evitar la dependencia sobre ninguna neurona en concreto, dado el factor de aletoriedad, se consigue un sistema mas robusto.
Otro objetivo de este proceso es evitar el sobre entrenamiento, que se da cuando se obtienen resultados mucho mejores en la etapa de entrenamiento, que en la etapa de test, debido a que el sistema se adapta demasiado a la base de datos de entrenamiento.

2.5 Procedimiento

- Generación y particionado de la base de datos

El primer paso es decidir que tipo de dato de entrada al sistema vamos a utilizar y como lo vamos a generar, a partir de una base de datos. Una vez definido, se tienen que escoger diferentes particiones en la base de datos para conseguir entrenar de una forma adecuada la red. Una mala elección de estas particiones podría resultar en una mala generalización y un mal entrenamiento de la red. Toda base de datos para entrenar debe tener al menos dos particiones: entrenamiento y test. Aproximadamente la partición de entrenamiento debe tener un 70% - 80% de los casos y el resto la de test. En algunos casos se añade una partición extra que divide la partición de entrenamiento, llamada partición de validación. Esta partición suele ser de un 25-30% de la partición de entrenamiento. La partición de validación es utilizada para evaluar (normalmente después de cada época) el entrenamiento que se ha llevado a cabo con los sujetos de entrenamiento y utilizar esta información para optimizar los valores de los filtros de la red.

- Entrenamiento

El entrenamiento es la etapa por la que pasa cualquier red para aprender todos sus pesos hasta clasificar los datos de entrada. Este proceso es uno de los más complejos y costosos en la creación de una red puesto que es dónde se computarán iterativamente todos los valores de los pesos de los filtros. Para empezar con el proceso, una vez creada la red, se deben inicializar los pesos con algún valor para poder empezar a iterar el algoritmo y empezar a encontrar los pesos idóneos para nuestra entrada. Seguidamente, todos los sujetos de la partición de entrenamiento serán utilizados como entrada para clasificarlos uno a uno. Con la clasificación obtenida, se compara con la clasificación real o deseada, para tener para tener unos valores con los que saber que precisión ha tenido esta iteración de entrenamiento. En este momento se aplica un algoritmo (en el caso de este proyecto, es el denominado Backpropagation) para mejorar los pesos de las capas de la red para conseguir una mejor clasificación. Este proceso se itera para cada sujeto de la base de datos de entrenamiento. Al proceso de entrenar toda la base de datos de entrenamiento, clasificar y actualizarlos pesos de las capas, se le denomina Época. Así que se entrena durante varias épocas, para ir mejorando el resultado obtenido.

- Backpropagation

Backpropagation es el proceso de optimización de los pesos de los filtros de la red. Es uno de los procesos mas importantes de el entrenamiento, este proceso calcula la desviación obtenida respecto a la clasificación real y actualiza los valores de estos filtros para minimizarlo mediante un algoritmo de optimización realizando el proceso desde las capas finales de la red, hacia las iniciales. Como anteriormente se ha comentado, este algoritmo se basa en aplicar la regla de la cadena para calcular de una forma más eficiente los gradientes del algoritmo de optimización. Estos cálculos de gradientes se efectúan de capas finales hacia el principio de la red, de aquí viene su nombre. En este proyecto se utiliza el método de optimización conocido como gradiente descendente estocástico (SGD), que calcula el error de la iteración, la dirección del gradiente que minimiza ese error y lo utiliza para actualizar los pesos.

3 Estado del arte

Como se ha comentado anteriormente, la tecnología de la redes neuronales, esta constantemente evolucionando y ampliando su metodología y radio de acción.
En este apartado comentaremos diferentes métodos de aplicación de las redes neuronales,herramientas para generar los programas necesarios y diferentes trabajos relacionados con el estudio de las nubes de puntos.
En realidad, esta tecnología ya se empezó a implementar entre los 80 y los 90, pero la limitación tecnológica (procesadores poco potentes y bases de datos demasiado pequeñas) no le permitieron encontrar un ámbito en el que asentarse y ser utilizado.
Actualmente con las mejoras de las CPUs y sobretodo la llegada de las GPUs, juntamente con la capacidad de almacenar grandes cantidades de datos, han hecho mejorar sus prestaciones y por lo tanto, ha aumentado su utilización.

3.1 Trabajos relacionados

En la actualidad, dado que la utilización de el formato nube de puntos ha aumentado su utilización, a causa sobretodo, de muchas aplicaciones relacionadas con la realidad virtual, y el hecho de que las redes neuronales ofrecen un abanico muy amplio para abordar los diferentes problemas de clasificación consecuentes, nos podemos encontrar con una gran variedad de métodos y técnicas diferentes para resolver este tipo de problemas. En este apartado se comentaran brevemente algunas técnicas.

3.1.1 SEGCloud:Semantic Segmentation of 3DPoint Clouds [1]

El objetivo de este articulo es realizar una clasificación y diferenciación de los diferentes objetos que se puedan encontrar en una escena. Para ello inicialmente voxeliza una nube de puntos, teniendo en cuenta los valores RGB y de ocupación de estos, para después introducir estos conjuntos de voxeles en una red neuronal formada por capas convolucionales y fully connected entre otras, de manera que cada voxel introducido, es clasificado a la salida de la red neuronal

3.1.2 Real-Time Plane Segmentation and Obstacle Detection of 3D Point Clouds for Indoor Scenes[2]

En este articulo, vuelven a realizar una voxelización de una nube de puntos, con el objetivo de disminuir el ruido (ya que para este caso un voxel se ve

menos afectado que un píxel) y así poder separar en un primer paso lo grandes planos, y posteriormente, realizar una segmentacion de los diferentes objetos.

3.1.3 Voxel and graph-basede point cloud segmentation of 3D scenes using perceptual grouping laws[3]

En este articulo, vuelven a utilizar una voxelización de una nube de puntos, con el objetivo de formar grafs, que son conjuntos de voxeles cercanos interconectados entre ellos, que serán segmentados posteriormente.

4 Metodología

En este apartado repasaré cual ha sido la metodología seguida durante el desarrollo del proyecto.
El objetivo del proyecto, es la clasificación de parejas de puntos que denominaremos keypoints, de dos nubes de puntos distintas que corresponden a dos vistas diferentes de la misma escena, en los siguientes grupos:

- Corresponden el mismo punto en la realidad
- No corresponden el mismo punto en la realidad

Para ello utilizaremos Redes neuronales convolucionales 3D, con la peculiaridad de que las nubes de puntos iniciales, pasaran un proceso de voxelización previo, con el objetivo de trabajar con una estructura de datos mas simple perdiendo la mínima información posible. He dividido el proyecto en tres bloques:

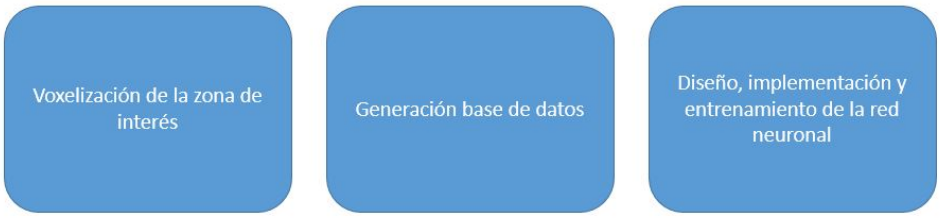

Figure 7: *Bloques que forman el proyecto*

4.1 Voxelización

Algo a definir en el sistema es la estructura de entrada que vamos a utilizar. En nuestro caso, hay que encontrar alguna manera de estructurar la nube de puntos, y más específicamente, la zona de alrededor de el punto de interés o keypoint.
Una técnica que se ha utilizado en varios artículos es la técnica de voxelización.
Un voxel es el equivalente en 3D de un píxel. Por lo que la Voxelixación, trata de estructurar una matriz de voxeles, con el objetivo de clasificar cada

punto de la zona de interés en el voxel correspondiente de la matriz de voxeles definida, para después unificar de alguna manera la información de los diferentes puntos contenidos en cada voxel.
Así el objetivo seria voxelixar la zona de interés de la nube de puntos, es decir, la zona del punto que queremos comparar y su alrededor, generando una estructura en la que para cada voxel, realizaríamos la media de los valores RGB, de los puntos que pertenecieran a cada voxel en concreto.
Esto genera diferentes dudas:

- Que tamaño de vóxel utilizar y con cuantos voxeles voxelizar la zona de interés?

Es un parámetro importante, ya que a causa de una mala elección de este parámetro, puede generar que haya demasiados puntos en un voxel, y se pierda demasiada información. O si escogemos demasiados voxeles de un tamaño demasiado grande, se tendría una estructura con demasiada información, que comportaría un mayor peso y por lo tanto mas tiempo de procesado.
Como en el momento en que se empezó a trabajar en la voxelización, no se disponía de las nubes de puntos con las que se entrenaría al sistema, por lo que se tomo la decisión de implementar tanto la opción de generar una voxelización tanto con un tamaño y número fijo de voxeles, como la opción de fijar un número de voxeles y definir un tamaño en función de cada nube de puntos.

- Que hacer con los Voxeles vacios?

El problema viene dado por la información relacionada con los valores RGB que contienen estos voxeles, ya que a un vóxel vacío, no seria correcto que sus valores de RGB fueran (0,0,0) ya que en realidad, esos valores equivalen al color negro y no seria correcto. Tampoco se le puede definir un valor nulo, ya que si trabajamos con redes convolucionales, es necesario que disponga de un valor, a parte de que supondría muchas dificultades en relación a la implementación.
Para solucionar este problema se ha decidido añadir un valor en la información del vóxel, que indique la ocupación del voxel, es decir: a parte de los valores RGB, añadiriamos un valor O, correspondiente a la ocupación del vóxel, que oscilaria entre 0 y 1 y seria el resultado de dividir, el número total de puntos que corresponden al vóxel en cuestión, divididos entre el numero de puntos totales contenidos en la estructura.
El script a implementar debe tener como argumentos de entrada: la nube de puntos a voxelizar, el punto que se utilizará como centro de la matriz de

voxeles(de cara a la base de datos, será uno de los dos puntos que estamos comparando para saber si en la realidad son el mismo punto), la distancia que cubrirá la matriz y la cantidad de voxeles que tendrá la matriz. Las primeras pruebas se realizaron con la nube de puntos bunny generada por la universidad de Stanford. En las siguientes imágenes se puede visualizar primero la forma de la nube de puntos, seguida de la nube de puntos voxelizada con una matriz de 10x10x10 voxeles, donde se observa que se puede reconocer perfectamente la forma, y una tercera figura en la que con una escala de color, se diferencia la ocupación de cada voxel.

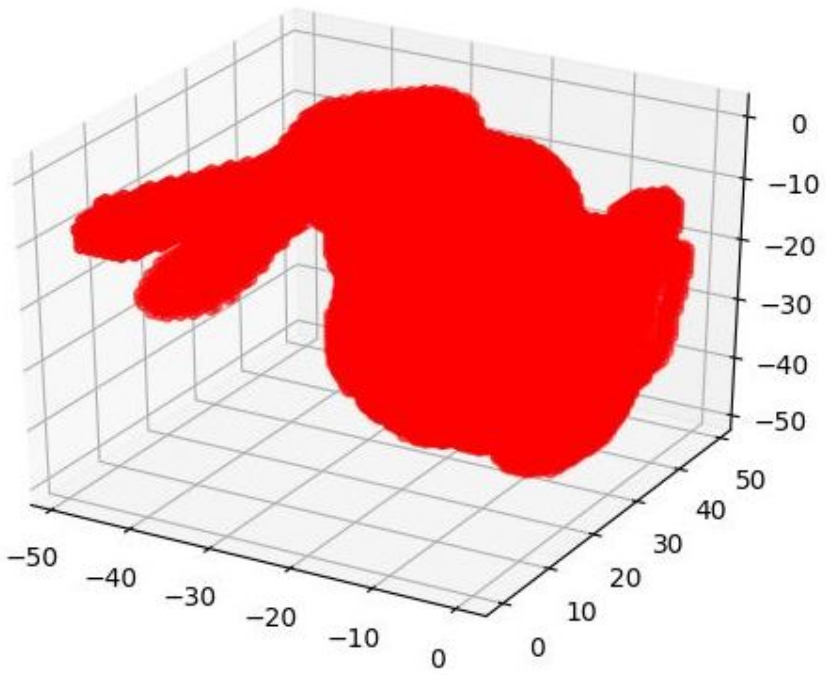

Figure 8: *Nube de puntos*

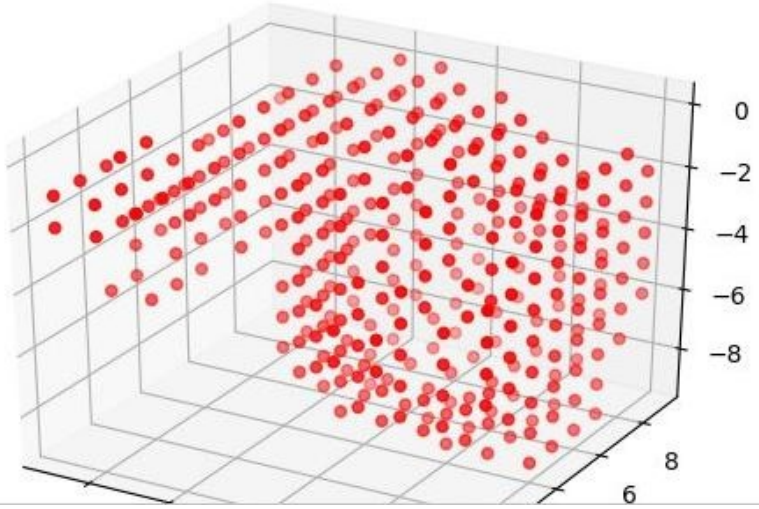

Figure 9: *Voxelización 10x10x10*

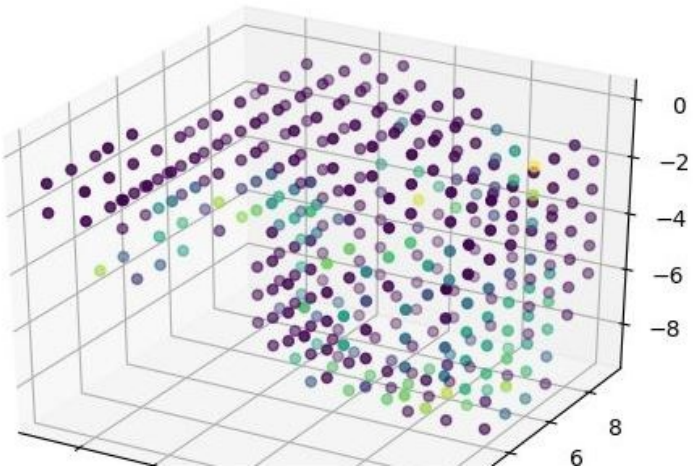

Figure 10: *Ocupación en cada voxel*

4.2 Generación de la base de datos

Para este proyecto, era necesario una base de datos que a parte de incluir una gran cantidad de nubes de puntos, incluyera un listado de parejas de puntos o keypoints, que pertenezcan a dos nubes de puntos diferentes de la misma escena, indicando si en la realidad son o no el mismo punto.

Esta base de datos, ha sido proporcionada por Alba Pujol, que ya tenia creada para sus anteriores proyectos, con la información necesaria, y información añadida sumamente útil (como podremos ver mas adelante).

La información de los keypoints, aplica al dataset Freiburg, que son un conjunto de nubes de puntos de diferentes escenas,captadas en la universidad de Munich, que contienen la información que necesitamos que este disponible en cada nube de puntos, es decir, información de los componentes RGB-D que corresponde a la información de los componentes de color mas información de profundidad.
Entonces el objetivo es, utilizando el script definido en la sección anterior, crear una estructura que contenga toda la informacion de las matrices voxelizadas, para cada pareja de keypoints, junto con un array, que indique si cada pareja de keypoints corresponden al mismo punto de la escena de la que forman parte las dos nubes de puntos a las que pertenecen los keypoints, o no.
Para generar esta estructura, había que decidir dos parámetros clave: La cantidad de voxeles por matriz y la distancia de la nube de puntos, alrededor del keypoint, que seria cubierta por la matriz. La elección de la cantidad de vóxeles, venia definido por un compromiso, un compromiso entre cantidad de información y peso de la estructura. Si se genera una estructura con muy pocos voxeles, toda la información seria muy comprimida lo que haría que la estructura no ocupara mucho espacio en memoria, y por lo tanto, agilizara el proceso de entrenamiento, pero probablemente al comprimir demasiado la información, esta se perdería información, tanto espacial como información RGB. En cambio si generamos una estructura demasiado grande, resultaría en una baja ocupación de los voxeles, por lo que la información estaría muy repartida y perderíamos muy poca información, pero a cambio de que la estructura ocupe mucho mas espacio en memoria y el proceso de entrenamiento fuera mucho mas lento.
El valor escogido ha sido de 16x16x16, ya que es un valor que asegura no perder demasiada información espacial ni información de valores RGB, pero si que es cierto que esta seguridad es dada a cambio de una base de datos un poco pesada.
En cuanto a la distancia de la nube de puntos a cubrir por la matriz de voxeles, era un valor critico, por que esta directamente relacionado con la cantidad de información que introducimos en una matriz, y vuelve a generar un compromiso parecido al caso anterior, es decir una relación cantidad de información-tiempo de procesado, que hay que tener en cuenta. El problema de este parámetro es que hay muchas posibilidades diferentes y que es difícil

tomar un valor orientativo, así que se ha decidido utilizar un valor, que ya había sido comprobado en otros proyectos que es un valor adecuado.
Concretamente el valor procede de la información suministrada por Alba Pujol, valor que ella ha utilizado en sus proyectos, y que es definido como 16 veces la distancia media de cada punto de la nube de puntos a un punto mas cercano.

4.3 Diseño y entrenamiento de la red neuronal

Para diseñar una estructura de la red neuronal, hay que definir primero en función detener en cuenta los datos que vamos a utilizar como punto de partida para marcar una serie de parámetros que nos ayudaran a definir la estructura.
La primera peculiaridad viene dada de que realmente, por cada iteración del proceso de entrenamiento, se van a tratar dos estructuras de voxeles diferentes. Al ser dos estructuras diferentes, que aunque sean el mismo punto en la realidad, pueden no tener las mismas características, es lógico utilizar una primera fase de la red neuronal, separada para cada una de estas estructuras de voxeles, y una segunda fase que trate la información que proviene de la primera fase, de estas dos vías separadas, una toda la información, y en esta segunda fase se trate como una sola vía. Para definir las capas de cada una de estas vias hay que definir los siguientes parametros:

- Numero de capas
- Cantidad de neuronas (o tamaño del filtro si es una capa convolucional)
- Características de salida

Al principio de la primera fase disponemos de dos estructuras de 16x16x16 con 4 características por cada una de las posiciones. El objetivo debería ser hacer cada vez la estructura menos ancha, pero mas profunda, es decir, mas características, en un tamaño mas pequeño de la estructura, por lo que cada una de las vías tendrá la siguiente estructura:

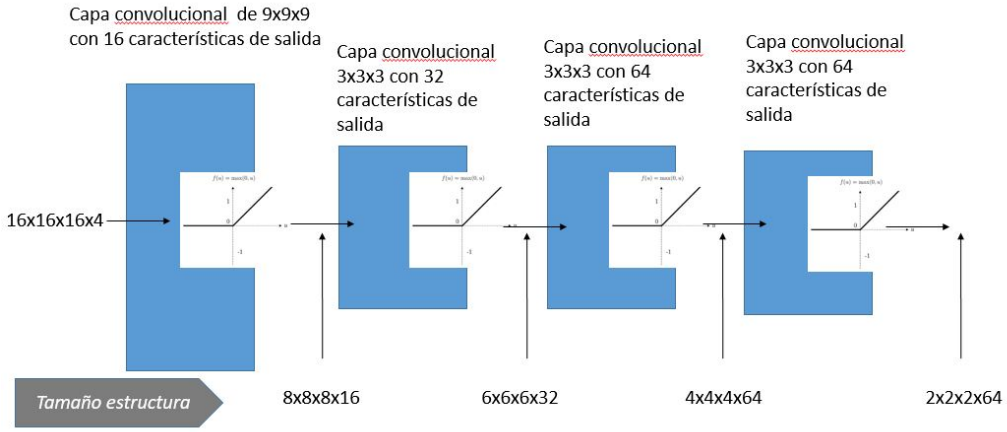

Figure 11: *Estructura de una rama*

Con dos ramas de este tipo en paralelo, obtenemos dos estructuras de información con la siguiente forma: 2 x 2 x 2 x 64.
A continuación, debemos unir estas dos estructuras de información y tratarlas como si solo fuera un flujo, por lo que concatenaremos todos los valores de estas dos estructuras de información, y utilizaremos capas fully connected, para obtener la clasificación binaria final:

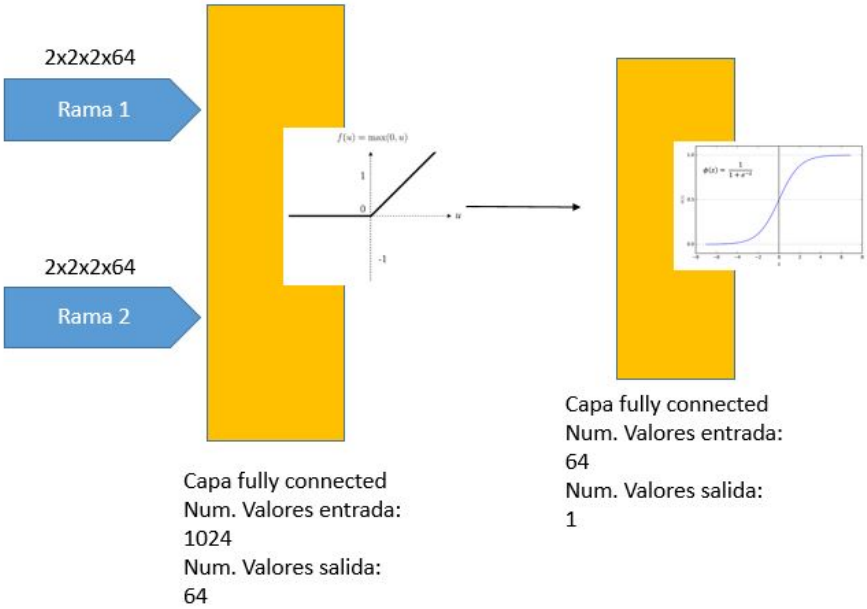

Figure 12: *Estructura fully connected*

De cara a computar las perdidas que se obtienen en el proceso de entrenamiento se utilizará la función de entropía cruzada binaria, en concreto ya utilizaremos una función del framework que incluye la implementación de la capa sigmoid, y en cuanto a la función de optimización, se utilizará la función de gradiente descendiente estocástico.

4.4 Herramientas implementación

En la actualidad hay muchos frameworks en diferentes lenguajes que ayudan a implementar, entrenar y testear un sistema de machine learning.
Previamente a este proyecto, todo lo relacionado con el machine learning y mas específicamente con las redes neuronales lo había implementando utilizando Matlab y una de sus toolbox particulares dedicada a ello. Aunque si que es cierto que actualmente hay frameworks mucho mas potentes y que dan acceso a mas opciones.
Los siguientes son los mas conocidos:

4.4.1 TensorFlow

Es la libreria que utiliza Google. Basada en C y Python. Es de las librerías mas rápidas y robustas, con la ventaja de que hay constantes actualizaciones resolviendo problemas y añadiendo funcionalidades.

4.4.2 Torch

Aunque esta basada en Lua, motivo por el que es bastante rechazado este framework, es también utilizado por grandes compañías internacionales. Proporciona una gran rapidez.
PyTorch, que es el framework utilizado en este proyecto, esta basado en Torch.
Entre otras razones, ha sido escogido este framework por las facilidades que aporta Python, en forma de librerías, a la hora de implementar. Por ejemplo, en el proyecto se utiliza la libreria Plyfile, que lee un archivo **.ply* (formato de las nubes de puntos) y devuelve una estructura, en la que indica: numero de puntos, posición y valores RGB de cada punto de la nube de puntos.

4.4.3 Keras

Librería basada en Python y R. Su peculiaridad es que trabaja en un nivel mas alto que TensorFlow o Torch. Por esta razón, aunque proporcione menos opciones que TensorFlow o Torch, es una libreria mas agil y que minimiza el codigo escrito.

5 Resultados

En esta sección se realiza un estudio de los resultados obtenidos y se explica el proceso de adaptación de la red, mediante diferentes parámetros, con el objetivo de mejorar los resultados obtenidos.
Para realizar estas pruebas, se han utilizado diferentes datasets que pertenecen a la base de datos de la universidad de Munich, los cuales estan repartidos en nueve secuencias, las cuales se utilizaran sobre la red neuronal que se muestra en la figura:

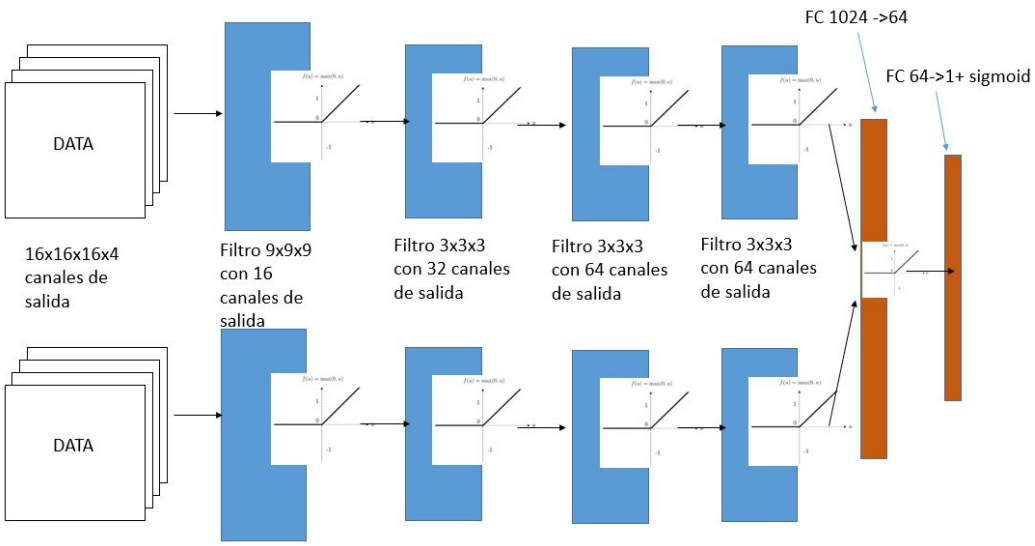

Figure 13: *Estructura completa*

En total se dispone de 325087 parejas de keypoints, que han sido divididos en el dataset de entrenamiento, el dataset de validación, y el dataset de test de la siguiente manera:

Dataset	Muestras totales	Secuencias	Porcentaje
Entrenamiento	175276	1, 5	54%
Validación	79398	2, 3, 8	24,40%
Test	70413	4, 6, 7, 9	21,65%

Cada muestra esta compuesta de dos matrices de 16 x 16 x 16 posiciones, en la que cada posición tiene los datos RGB + ocupación, recogidos como se ha indicado anteriormente.
En las gráficas que se mostraran, mayoritariamente se mostraran los datos de precisión y la funcion de perdida obtenidos en cada prueba. Mediante

la precisión (accuracy) medimos cuantos resultados hemos clasificado bien respecto todos los valores clasificados mientras que la funcion de pérdida, mide la diferencia entre la clasificacion realizada, y la clasificacion real.
Definida la base de datos, las primeras pruebas se realizaron con el objetivo de comprobar que la red neuronal implementada efectivamente se entrenaba a medida que se introducían los datos.
Asi que se realizo una prueba con una cantidad pequeña de épocas, y una sola secuencia como dataset de entrada (con diferente formato al que sera utilizado en las siguientes pruebas), sin realizar el proceso de validación. Estos fueron los resultados:

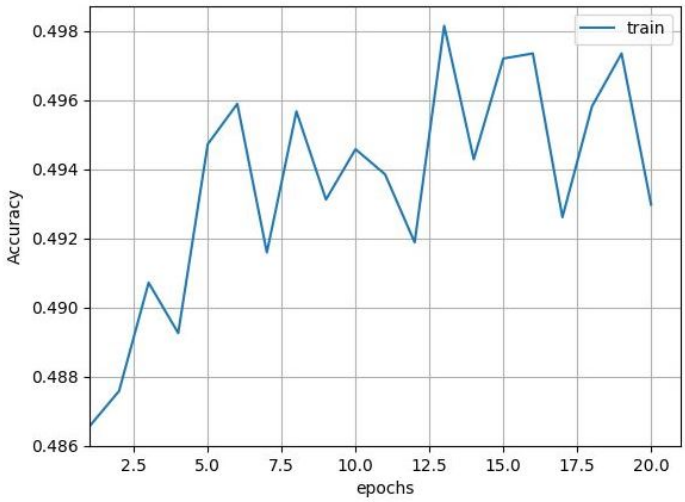

Figure 14: *1a Prueba*

Como se puede observar, aunque los resultados no acaban de ser buenos, el sistema, a medida que se realizan épocas, va mejorando el resultado. A continuación el objetivo era intentar que la red llegara a una situación de sobre entrenamiento con un dataset mas grande, para comprobar que con la información de entrada, la red no se estancaba. Para conseguir esto, aumentaremos considerablemente la cantidad de épocas durante las que entrenaremos el sistema y deberiamos observar, en la curba de precisión, como en el tramo final, se obtienen valores cercaos al 100% de acierto, pero que esta mejora no se refleja en el proceso de validación.

Y efectivamente, podemos observar que aunque la gráfica de la precisión, indique que los resultados en el entrenamiento son buenos, cercanos al 100%

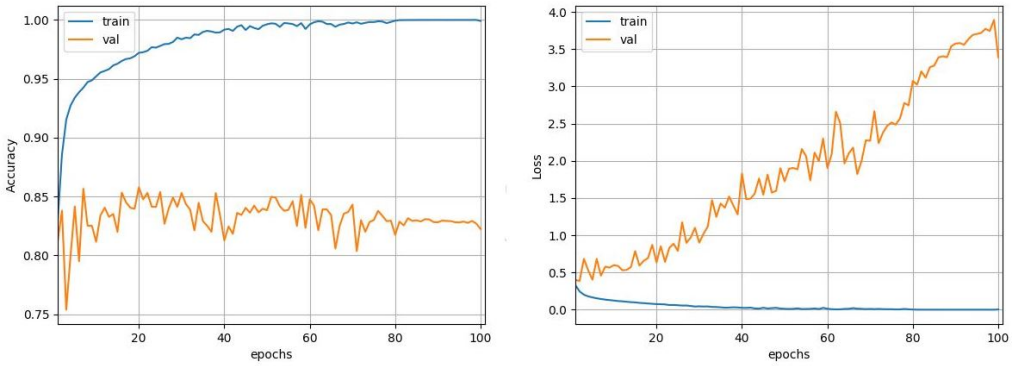

Figure 15: *Test sobreentrenamiento*

de acierto, en cuanto a la validación, llega un punto que no mejora y que cada vez aumenta la función de pérdida.
Ahora que ya estamos seguros que el comportamiento de la red es el correcto, vamos a iniciar el proceso de modificación de los parámetros learning rate y weight decay con la intención de buscar la combinación que mejor resultados da. Para esto, primero realizamos 3 pruebas con 3 valores diferentes de learning rate, y una vez detectado el mejor valor pasaremos a probar diferentes valores de weight decay. Como observamos en estas dos pruebas realizadas,la gráfica de la izquierda es una prueba realizada con un valor de learning rate de 0.0001 y la de la derecha con un valor de 0.00001 y efectivamente al disminuir el valor de learning rate, se suaviza la curva de precisión, y el sistema tarda mas épocas en llegar al sobre entrenamiento:

Se han realizado pruebas con3 valores distintos de learning rate, y diferentes cantidades de épocas, para que puedan llegar al punto en el que por mucho que entrenen, no se mejoren las prestaciones, y se han obtenido los siguientes resultados:

learning rate	20 Épocas	40 Épocas	60 Épocas
0,0001	0,8363		
0,00005	0,8304	0,8317	
0,00001	0,8347		0,8346

Como podemos observar, el mejor resultado obtenido, ha sido cuando se ha utilizado un learning rate de 0.0001. Como se ha comentado, se han realizado

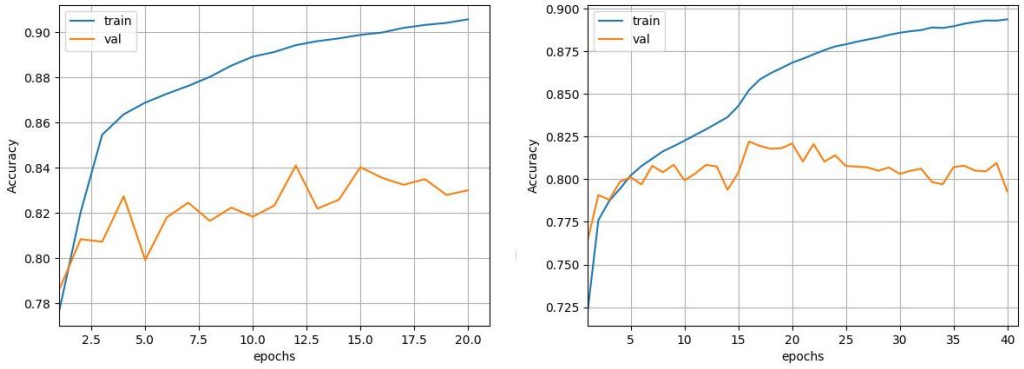

Figure 16: *Test curba learning rate*

pruebas con más épocas, para los otros dos casos, con la peculiaridad de que para el caso de 0.00001, el mejor resultado es el mismo, ya que es u resultado que se obtiene antes de llegar a las 20 épocas. También hay que indicar, que estos resultados, son los resultados obtenidos con los datasets de validación. Con este parámetro fijado, el siguiente paso es realizar el mismo proceso, pero con 3 valores de weight decay, aun que esta vez, si que se utilizará el dataset de test.
En este caso, se realizará entrenamientos de 50 épocas, y con los pesos que se obtengan el mejor resultado de validación, Se realizara una clasificación sobre el dataset de test, y se calculará la precisión:

	Weight decay 1		Weight decay 0,1		Weight decay 0,01	
learning rate	Validation	Test	Validation	Test	Validation	Test
0,0001	0,7931	0,7831	0,7985	0,7871	0,8402	0,8027

De manera que obtenemos que la mejor configuración para la red utilizada, es con un learning rate de 0.0001 y un weight decay de 0,01. A continuación, las pruebas realizadas, están dirigidas a comprobar que si al aplicar un proceso de dropout, y un proceso de ampliación de datos, mejoran los resultados sobre la misma red neuronal. En el caso de la ampliación de datos, de entre todos los métodos que se pueden utilizar, se ha probado un método que cambia aleatoriamente la iluminación de unos datos de entrada escogidos a partir de cierta probabilidad definida (en este caso el 50%)

	Data augment.		Dropout	
learning rate+wd	Validation	Test	Validation	Test
0,0001//0,01	0,8265	0,8203	0,7273	0,6967

En ninguno de los dos casos, obtenemos resultados que mejoren lo que se había conseguido anteriormente. Pero para el caso del dropout, podría ser interesante como configuración de cara a realizar pruebas, dado que utilizando estas capas, se disminuye mucho el tiempo de entrenamiento.
Finalmente, se han realizado una serie de pruebas, con el objetivo de ver como podría afectar, utilizar una base de datos de diferente tamaño a la que se ha escogido. A partir de la base de datos que ya tenemos creada, se han generado dos bases de datos, que difieren en la cantidad de voxeles que se han utilizado al generar el vecindario, utilizando cubos de 8x8x8 y cubos de 4x4x4. Para poder realizar estas pruebas, ha sido necesario modificar el tamaño y la cantidad de los filtros convolucionales 3d. A continuación se puede ver la red neuronal utilizada en el caso de la base de datos con cubos de 8x8x8 voxeles y los resultados que se han obtenido:

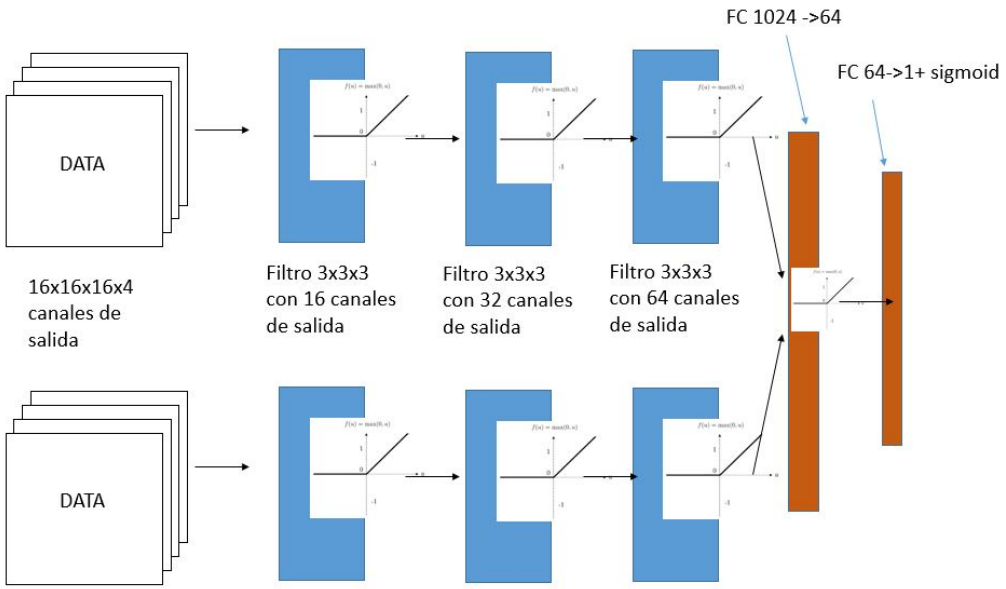

Figure 17: *Estructura 8x8x8*

	8x8x8	
learning rate+wd	Validation	Test
0,0001//0,01	0,8658	0,8176

Como podemos observar, para una red bastante parecida, utilizando la misma configuración de parámetros, esta red obtiene mejores resultados que la nuestra, además de realizar todo el proceso con bastante mas rapidez, en concreto unas 5 veces más rápido
Y a continuación, de la misma manera, los resultados con una base de datos con cubos de 4x4x4 voxeles:

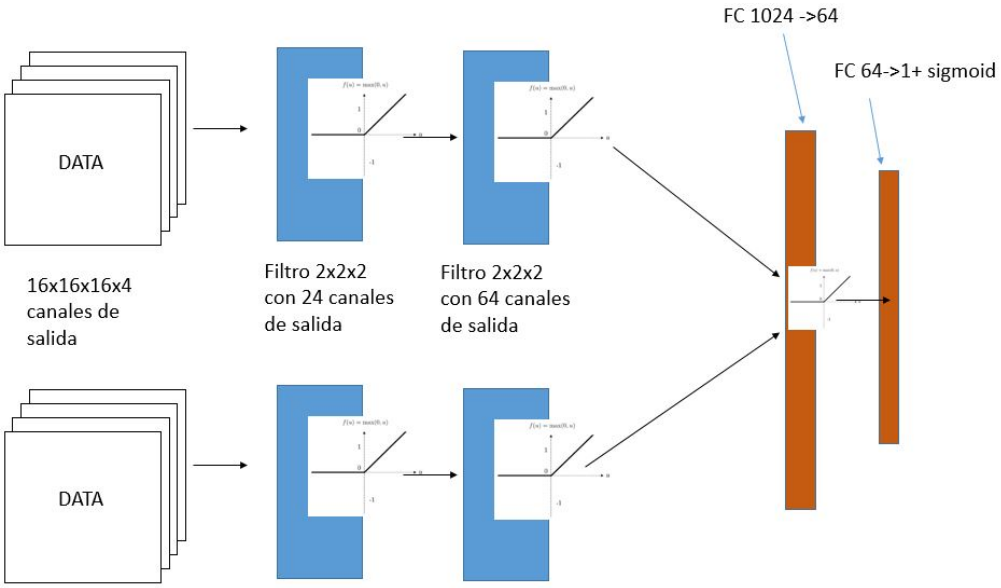

Figure 18: *Estructura 4x4x4*

	4x4x4	
learning rate+wd	Validation	Test
0,0001//0,01	0,8062	0,7211

En este caso, aunque la red neuronal difiere mas que el caso anterior y sigue disminuyendo aún mas el tiempo necesario para realizar el proceso, los resultados no son tan buenos como en los casos anteriores.

6 Presupuesto

En esta sección se analizará de forma teórica cual sería el presupuesto del proyecto para poderse llevar a cabo contando el material utilizado para trabajar y las horas empleadas en el desarrollo entero del proyecto. Dado que el producto final esperado del proyecto es un software en cuanto a coste material solo se añadirán las licencias de los programas y librerías necesarios. El lenguaje de programación utilizado ha sido Python y todas las librerías son de código abierto, por tanto, no incrementan el presupuesto final del proyecto, pero el IDE utilizado ha sido PyCharm de JetBrains y la licencia profesional tiene un coste de 20/mes, multiplicado por 7 meses de utilización del IDE, incrementa el coste 140. Un gasto a cubrir es el coste de las horas dedicadas al proyecto por el estudiante. A este proyecto se han dedicado unas 400 horas de desarrollo, mas unas 25 horas de tutorías y unas 100 horas de estudio, preparación y documentación. En total 525 horas de trabajo, que si lo multiplicamos por el salario estándar de un becario definido por la Universidad Politécnica de Catalunya (8/hora) resulta un coste de 4200
Al tratarse de un proyecto de investigación, desde el ámbito académico no se incluyen los costes del servidor de computación utilizado, el cual ha sido prestado por la UPC para llevar a cabo este tipo de actividades académicas. Por lo que, si sumamos estos valores, el presupuesto total del proyecto asciende a 4340 totales.

7 Conclusiones

Segun los resultados obtenidos en las diferentes pruebas realizadas en apartados anteriores, Este sistema obtiene buenos resultados de clasificación para esta base de datos
Comparando con otros proyectos y observando los resultado obtenidos se pueden definir ciertos ámbitos en los que se podria trabajar para obtener mejores resultados:

- Variación de la arquitectura: En este proyecto, exceptuando los casos en los que se ha cambiado el tamaño de la base de datos, solamente se ha utilizado un modelo de arquitectura. Hay una gran cantidad de tipos de capas, que generan una gran cantidad de combinaciones posibles por lo que probar otros tipos de arquitecturas parece un camino que pude llevar a mejores resultados, igual que aplicar i variar diferentes parámetros a los que se han probado en este proyecto.

- Diferentes tamaños de voxelización: Como se ha podido observar en las pruebas realizadas, al cambiar el numero de voxeles utilizados a 8x8x8, se han obtenido mejores resultados, basandonos en la estructura utilizada para cuando se genera una base de datos con matrices de 16x16x16. Asi que si se realizara un estudio al respecto, y se optimizara el tamaño de la matriz voxelizada, es muy probable que se mejoren los resultados obtenidos y se disminuya el tiempo de procesado.

Aún asi, los resultados obtenidos son muy positivos, y con los conceptos que se han adquirido son una base sobre la que cimentar futuros proyectos relacionados con las redes neuronales, ámbito que crece y seguira creciendo a una velocidad muy grande, lo que seguramente abrirá más opciones y alternativas que aplicar a este tipo de proyectos.

8 Bibliografia

[1] Lyne P. Tchapmi, Christopher B. Choy, Iro Armeni,JunYoung Gwak, Silvio Savarese,"SEGCloud: Semantic Segmentation of 3D Point Clouds"

[2]Zhe Wang, Hong Liu, Yueliang Qian, and Tao Xu:" Real-Time Plane Segmentation and Obstacle Detection of 3D Point Clouds for Indoor Scenes"

Y. Xu, L. Hoegner, S. Tuttas, U. Stilla"Voxel and graph-basede point cloud segmentation of 3D scenes using perceptual grouping laws"

[4] Javier de la Rica, Javier Ruiz Hidalgo "3D Ppoint Cloud correspondences using deep learning"

[5] Alba Pujol-Miro, Javier Ruid-Hidalgo, Josep R. Casas, Registration of Images to Unorganized 3D Point Clouds using Contour Cues, Image Processing Group, Universitat Politècnica de Catalunya