Universidad del País Vasco
Euskal Herriko Unibertsitatea

Máster Universitario en

Ingeniería Computacional y Sistemas Inteligentes

Departamento de Ciencias de la Computación e Inteligencia Artificial

Tesis de Máster

Estudio comparativo de diferentes aproximaciones a la clasificación de modelos 3D mediante Deep Learning

Aitor Ruiz de Samaniego López

Tutores

Basilio Sierra

Departamento de Ciencias de la Computación e Inteligencia Artificial

Facultad de Informática

Íñigo Mendialdua

Departamento de Lenguajes y Sistemas Informáticos

Escuela de Ingeniería de Bilbao

KISA ICSI

KZAA /CCIA

Master eta Doktorego Eskola
Escuela de Máster y Doctorado
Master and Doctoral School

Octubre 2020

Master eta Doktorego Eskola
Escuela de Máster y Doctorado
Master and Doctoral School

Índice

1 Introducción....4
1.1 Motivación....4
1.2 Resumen....4
2 Estado del arte....5
2.1 Estado del arte en la clasificación de imágenes 3D....5
2.1.1 3D ShapeNets: A Deep Representation for Volumetric Shapes [1]....5
2.1.2 Inductive Multi-Hypergraph Learning and Its Application on View-Based 3D Object Classification [2]....6
2.1.3 RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints [3]....6
2.1.4 PointNet: A 3D Convolutional Neural Network for Real-Time Object Class Recognition [4]....7
3 Técnicas utilizadas....8
3.1 Voxelización....8
3.2 Histogramas....9
3.3 Binarización de clases....10
3.4 Red neuronal....11
3.4.1 Neurona....11
3.4.2 Función de activación....12
3.4.3 Estructura de la red neuronales....13
3.4.4 Función de coste....14
3.4.5 Propagación hacia atrás....15
3.4.6 Descenso de gradiente....15
3.4.7 Learning rate....16
3.5 Red neuronal convolucional....17
3.5.1 Convolución....18
4 Experimentos....20
4.1 Hardware....20

4.2 Software....20
4.3 Preprocesado de los modelos: Voxelización....21
4.4 Parámetros del entrenamiento....22
4.4.1 Épocas....22
4.4.2 Early Stop....22
4.4.3 Cross validation....23
4.5 Configuración de las redes....23
4.5.1 Red Neuronal Densa....23
4.5.2 Convolución 3D....23
4.5.3 Histogramas....25
4.5.4 Red neuronal densa + One vs. One....25
4.5.5 Conv3D + One vs. One....25
4.5.6 Histogramas + One vs. One....25
4.6 Resultado obtenidos....26
4.6.1 Comparación por clase....27
5 Conclusiones y futuros trabajos....30
6 Bibliografía....31

1 INTRODUCCIÓN

En este trabajo se ha realizado una evaluación de diferentes métodos para la clasificación de modelos 3D. Se han aplicado métodos de clasificación supervisada sobre diferentes descriptores extraídos de mallas de modelos 3D.

1.1 MOTIVACIÓN

En los últimos años, la clasificación de modelos 3D se ha convertido en esencial para muchos aspectos de tecnologías que actualmente están más presentes en la sociedad, como la creación de nuevos modelos a partir de otros existentes para videojuegos o aplicaciones de realidad aumentada, o la identificación de objetos a partir de los nuevos sensores que se están incluyendo en todo tipo de dispositivos, desde vehículos autónomos hasta teléfonos móviles. En este trabajo, se pretenden evaluar distintos métodos de clasificación de modelos 3D basados en técnicas de machine learning, en concreto en métodos de clasificación supervisada, las metodologías que más se están actualizando actualmente.

1.2 RESUMEN

En este trabajo se han comparado la clasificación de modelos 3D mediante la extracción de diferentes características de los modelos. Primero se ha llevado a cabo una clasificación basada en la voxelización de los modelos, similar a la del estudio llevado a cabo el trabajo original de Modelnet [1] de la universidad de Priceton. En segundo lugar, se ha extraído un histograma de la ocupación del espacio en cada una de las dimensiones de los modelos y se ha realizado una clasificación con estos histogramas como input.

Para la implementación del primer caso se ha comparado una clasificación basada en una red neuronal densa y una red convolucional tridimensional, mientras que en el caso de los histogramas, se ha utilizado una red convolucional de 2 dimensiones.

Por último, en cada uno de los casos, se ha aplicado técnica de binarización de clases y se han comparado los resultados obtenidos.

2 ESTADO DEL ARTE

Para establecer un contexto adecuado en el presente trabajo, en este apartado se explican algunas de las aproximaciones actuales a la clasificación de modelos 3D.

Mientras que para la clasificación de imágenes 2D se cuenta con multitud de software e incluso con servicios que permiten la extracción casi inmediata de información de la imagen, para la clasificación de modelos 3D la mayoría de los trabajos encontrados son puramente académicos o resoluciones a problemas puntuales, muy alejado del nivel de servicios que se ofrecen para el tratamiento 2D. Gran parte de estos trabajos se recopilan en la propia página de Modelnet[1] cuyo dataset sirve de conjunto de pruebas y ranking para establecer la bondad de dichas metodologías.

2.1 ESTADO DEL ARTE EN LA CLASIFICACIÓN DE IMÁGENES 3D

2.1.1 3D ShapeNets: A Deep Representation for Volumetric Shapes [1]

Además de ser la base sobre la que se ha realizado el presente trabajo, sirve de punto de referencia a los demás investigadores para evaluar la validez de sus trabajos, estableciendo un ranking para la clasificación de 40 clases distintas de objetos 3D. Aunque los resultados de la clasificación obtenidos no son los mejores del ranking que la propia página del proyecto publica (un *tasa de acierto* del 77% en la clasificación de 40 clases) la importancia de los resultados obtenidos para esta tesis hace necesario que se desarrolle la metodología que se llevó a cabo para la obtención del dataset utilizado.

Se realizó una búsqueda en 261 sitios de internet dedicados a la clasificación de modelos CAD. En ellos se consultaron las categorías más comunes y se eliminaron aquellas que no tenían más de 20 resultados, resultando un total de 660 categorías. Se eliminaron los resultados clasificados incorrectamente de forma manual utilizando el servicio Amazon Mechanical Turk, un servicio que mostraba a un grupo de personas contratadas un modelo 3D y debían contestar si pertenecía o no a una categoría concreta. Después, se eliminaron los elementos irrelevantes del modelo, como el

1 http://modelnet.cs.princeton.edu/

suelo, personas cerca del objeto etc., de tal forma que en modelo solo contuviese un objeto que perteneciese a la categoría etiquetada. Se eliminaron modelos poco realistas (objetos demasiado simplificados, aquellos que solo contenían imágenes del modelo) y modelos duplicados. Finalmente se obtuvieron 151.228 modelos CAD que pertenecían a 660 categorías únicas. De estas se seleccionaron 40 para llevar a cabo la clasificación, y son las que actualmente sirven para establecer el mencionado ranking (Modelnet40).

2.1.2 Inductive Multi-Hypergraph Learning and Its Application on View-Based 3D Object Classification [2]

Este ha sido de los métodos que mejores resultados ha obtenido en la clasificación de la base de datos Modelnet40, obteniendo una *tasa de acierto (accuracy)* del 97,16%. Este método se basa en la aplicación de aprendizaje inductivo, es decir, la extracción de reglas generales a través de ejemplos, a hipergráfos generados a partir de los vértices de los modelos 3D. Los hipergráfos se contruyen relacionando los vértices del modelo con el resto de vértices a través de hiperaristas mediante kNN, conectando el vértice a los K vértices más cercanos. De este hipergráfo aplica el entrenamiento inductivo de forma recurrente para obtener el modelo de clasificación.

2.1.3 RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints [3]

Otra de las aproximaciones con mejores resultados registradas en Modelnet con un 97,37% de *accuracy* es la llamada RotationNet, cuyo enfoque principal es realizar la clasificación no solo del modelo de entrada, si no de un conjunto de *vistas* obtenidas a partir de la colocación de la "cámara" que observa el objeto en distintos puntos alrededor del objeto a clasificar. Así pues, este método convierte la clasificación de objetos 3D, en una clasificación de imágenes 2D, lo que permite un rendimiento mucho mayor que otros como la voxelización, sin pérdida de capacidad predictiva. Si bien este método puede resultar de utilidad en ciertos escenarios (con objetos virtuales o cuando sea posible obtener estas vistas 2D), puede no ser útil para casos como identificación de objetos en tiempo real en sistemas de conducción autónoma, donde difícilmente se podrían obtener vistas desde distintos ángulos.

2.1.4 PointNet: A 3D Convolutional Neural Network for Real-Time Object Class Recognition [4]

Este estudio parte de una base similar a Modelnet: aplicar una convolución 3D a un sistema voxelizado. Sin embargo, mientras que en Modelnet cada voxel del espacio representaba si esa región del espacio estaba ocupada o no por el modelo (0/1), en este trabajo, la malla que representa el objeto 3D se transforma en una nube de puntos, cada punto se mapea en un voxel del espacio y el resultado es una conjunto de valores para cada voxel que representa el número de puntos de la nube de puntos que se encuentran localizados en ese voxel. Tras este preprocesamiento de datos, se aplican una arquitectura de red neuronal con una arquitectura con dos capas convolucionales seguidas de pooling y una capa densamente conectada antes de las salidas. Con esta configuración se obtiene una precisión del 77.6% en una clasificación con 10 clases (conjunto Modelnet10).

3 TÉCNICAS UTILIZADAS

Para la realización de los experimentos se han utilizado una serie de técnicas tanto de preprocesado de datos (voxelización e histogramas) como en el ámbito de la clasificación propiamente dicha (binarización, redes neuronales densa y convolucionales).

3.1 VOXELIZACIÓN

La voxelización es la técnica utilizada en el trabajo base en el que se creó la base de datos de evaluación utilizada. Otros trabajos [5] se han especializado en optimizar el voxelizado y alineamiento de la base de datos, pero aquí se ha preferido realizar el proceso como parte del trabajo. La técnica consiste en la segmentación del espacio ocupado por un modelo 3D en cubos o *voxels* para determinar si el modelo se encuentra total o parcialmente en esa zona del espacio. Para el presente trabajo, no se ha tenido en cuenta el nivel de ocupación de cada *voxel*, indicando únicamente si el modelo 3D se encuentra en dicho *voxel*. El espacio se ha dividido en una matriz tridimensional de 24x24x24, habiendo hecho experimentos con distintas resoluciones (32x32x32 y 48x48x48) sin obtener diferencias altamente significativas.

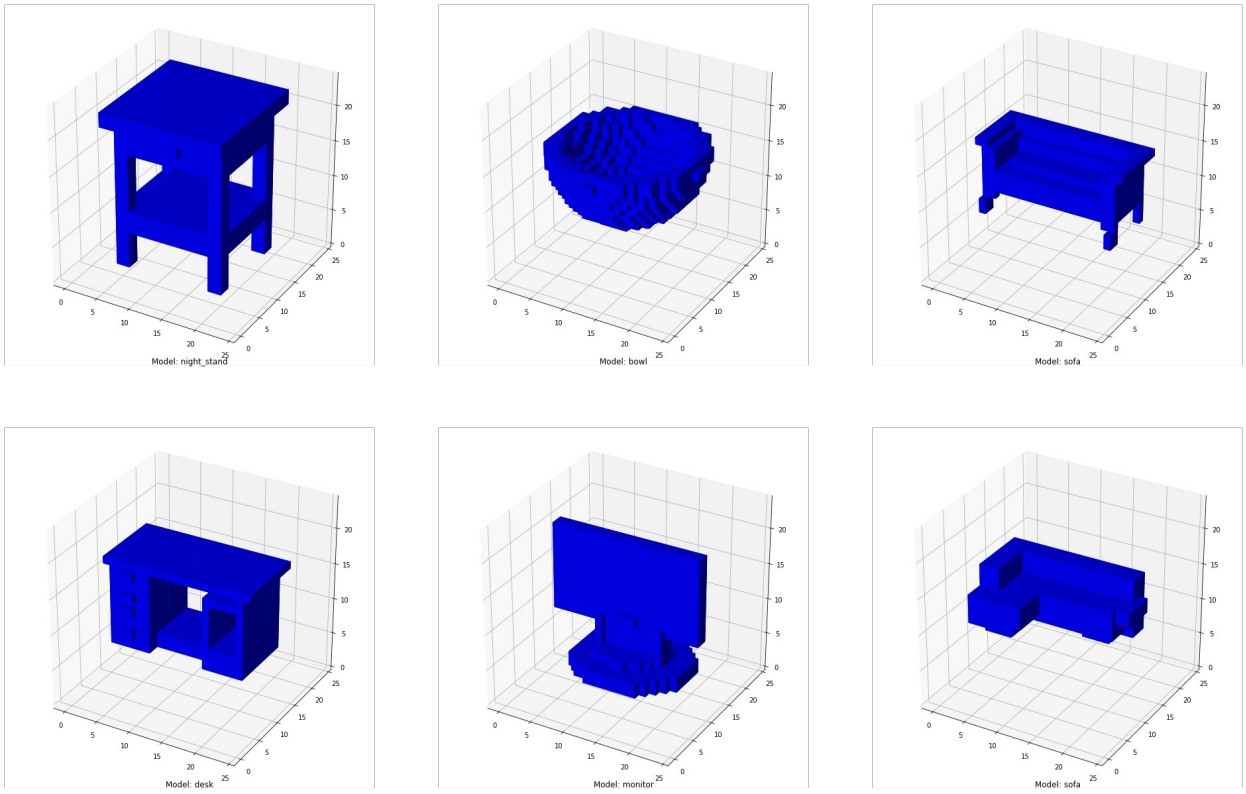

Figura 1: Voxelización de modelos

3.2 HISTOGRAMAS

Aplicando un paralelismo con las técnicas de clasificación de imágenes 2D que utilizan histogramas [6][7], se pretende utilizar técnicas semejantes para la clasificación de modelos 3D. Los histogramas utilizados representan la ocupación del modelo 3D del espacio que lo contiene. Para calcularlo, se han dividido cada una de las 3 dimensiones en 24 segmentos y se ha calculado cuantos de los segmentos de las otras 2 dimensiones están ocupados por el modelo.

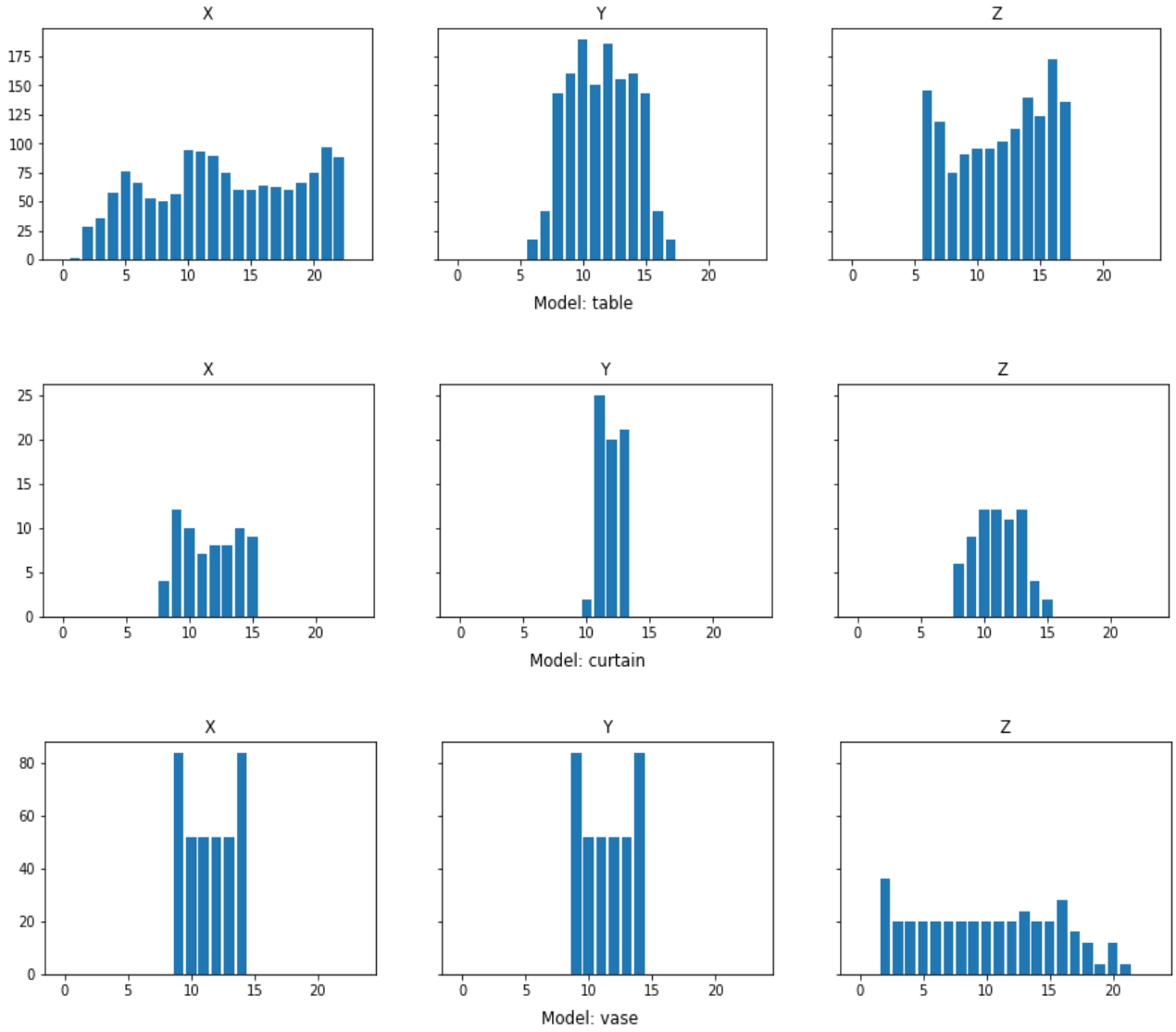

Figura 2: Histogramas de modelos

3.3 BINARIZACIÓN DE CLASES

Es una técnica que se basa en dividir un problema de clasificación multiclase en una serie de problemas de clasificaciones binarias. Esta técnica se puede aplicar de distintas maneras, como por ejemplo, enfrentando cada clase con el resto (one vs all), o enfrentando cada clase con cada una de las otras clases (one vs one). En este trabajo se utiliza la técnica one vs one ya que en general obtiene mejores resultados que one vs all [8].

Para ello se han entrenado cada una de las clases a clasificar contra el resto de clases, obteniendo así una serie de $\frac{C(C-1)}{2}$ clasificadores, siendo C el número de clases. Para clasificar una nueva instancia, ésta será clasificada por cada uno de los clasificadores obteniendo una serie de predicciones. Para tomar la decisión final, se suelen utilizar distintas técnicas que combinan estas predicciones, siendo el voto mayoritario una de las técnicas más utilizadas. Es decir, a la nueva instancia se le asigna la clase que más votos ha obtenido.

Existen algunos trabajos [9] [10] que reducen el número de clasificadores necesarios para realizar una clasificación más eficiente. Esto es debido a que en problemas donde el número de clases es muy alto, se hace poco eficiente el entrenamiento y ejecución de un número tan alto de modelos. En el caso que se trata aquí, el número de clases es 40 y el número de clasificadores creados es 780, lo cual es asumible en términos de capacidad de computo.

3.4 RED NEURONAL

Las redes neuronales artificiales son unos de los algoritmos de machine learning más utilizados hoy en día. Consisten en un conjunto de unidades, llamadas neuronas artificiales. Estas neuronas se agrupan por capas y las capas envían sus salidas a las capas siguientes. Las salidas de la capa final se interpretan como una clasificación o una regresión. Estos sistemas *aprenden* y se forman así mismos en lugar de ser programados de forma explicita modificando parámetros internos como se explica en las siguientes secciones.

3.4.1 Neurona

Una neurona artificial es un modelo matemático que intentan asemejarse al comportamiento de una neurona biológica. Cuando recibe una entrada, le aplica una combinación lineal, como resultado se obtiene que la neurona se activa o no. El modelo más simple se define mediante la siguiente función:

$$z = f_{act}\left(\sum_{i=1}^{n} w_i x_i + b\right)$$
$$y = \begin{cases} 1 & si\ z \geq 0 \\ 0 & si\ z < 0 \end{cases}$$

Donde *n* es el número de entradas de la neurona, *w* es el array de pesos (*weight* en inglés) y *b* es el sesgo (*bias* en inglés).

3.4.2 Función de activación

Existen más funciones para determinar la salida de una neurona. En el ejemplo anterior la neurona se *activaba* si el su resultado era mayor que 0 y no se activaba si era menor. Algunos otros ejemplos que se utilizan habitualmente para esta función son:

- Lineal: no modifica el resultado de la neurona.
- Sigmoide: Se corresponde con la función $f(x) = \frac{1}{1 - e^{-x}}$ y sus características son:
  - Satura el gradiente
  - Lenta convergencia
  - No está centrada en el 0
  - Está acotada entre 0 y 1
  - Buen rendimiento en la última capa
- Tangente hiperbólica: Se corresponde con la función $f(x) = \frac{2}{1 + e^{-2x}} - 1$ y sus característica son:
  - Satura el gradiente
  - Lenta convergencia
  - Centrada en el 0
  - Acotada entre -1 y 1
  - Se utiliza para decidir entre una opción y su contraria
- ReLu (Rectified Linear Unit): Se corresponde con la función $f(x) = max(0, x)$ y sus características son:

- Solo se activa para valores positivos
- No está acotada
- Se comporta bien con imágenes
- Se comporta bien en redes convolucionales

- Softmax: Transforma las salida a una representación en forma de probabilidades. Su función es $f(z)=\frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}}$ y sus características son:
  - Obtener una representación en forma de probabilidad
  - Acotada entre 0 y 1
  - Normalizar multiclase
  - Se comporta bien en las últimas capas

3.4.3 Estructura de la red neuronales

Las redes neuronales son conjuntos de neuronas agrupadas por capas, donde habitualmente cada neurona de cada capa, tiene como entrada todas las salidas de la capa anterior. La salida de cada neurona es entrada de todas las neuronas de la capa siguiente:

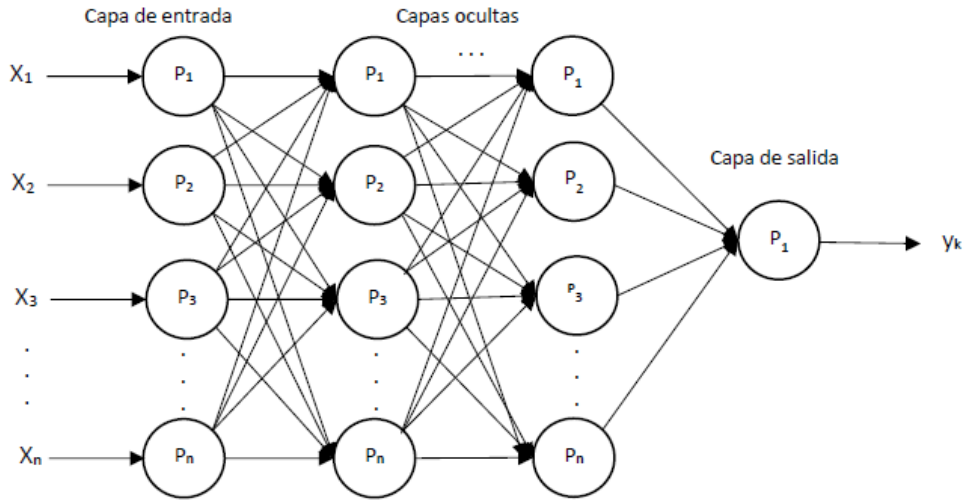

Figura 3: Arquitectura de red neuronal

La capa de entrada será un vector de números que representan las características de los modelos y la capa de salida representa las cualidades que se quieren calcular a través de la red neuronal. Esto puede ser un *score* para los casos de clasificación, que indicará con qué probabilidad un modelo concreto pertenece a una clase, o un valor real en el caso de regresión.

3.4.4 Función de coste

La función de coste es la forma que se utiliza para determinar cuan acertada es una clasificación, es decir, la diferencia entre el valor obtenido y el valor esperado, con el objetivo de optimizar los parámetros de pesos y sesgo. Se pueden utilizar distintos tipos de funciones de coste como son:

- Raíz cuadrada media (RMSE): Su función es:

  $$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i + y_i)^2}{n}}$$

  y sus características son:

  - Penaliza los valores grandes
  - Funciona bien para regresiones

- Error absoluto medio (MAE): Su función es:

  $$MAE = \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{n}$$

  y sus características:

  - Difícil diferenciación y convergencia
  -
  - Fácilmente interpretable

- Error absoluto medio escalado (MASE): Su función es:

  $$MASE = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{\frac{n}{n-1} \sum_{i=1}^{n} |\hat{y}_n - y_{n-1}|}$$

  y sus características:

- Más difícil diferenciación y convergencia
- Simétrica
- Fácil de interpretar

- Entropía cruzada binaria (Binary Cross-Entropy): Su función es:

$$\mathscr{L}(\theta)=\frac{-1}{n}\sum_{i=1}^{n} y_i \log(p_i+(1-y)\log(1-pi))$$

y sus características son similares a MASE.

3.4.5 Propagación hacia atrás

Propagación hacia atrás es la forma en la que los pesos y los sesgos de la red se van a actualizar durante el entrenamiento para hacer que el resultado de la red converja hacia los valores correctos. Para conocer cómo cada peso y sesgo de cada neurona afecta al resultado, se utilizan las derivadas parciales de la función de coste con respecto a ese peso o sesgo. Estas derivadas se utilizan para calcular los pesos y sesgos en la siguiente iteración a través del llamado descenso de gradiente.

3.4.6 Descenso de gradiente

Las derivadas de los pesos y sesgos de la función de coste, permiten tener una visión de cómo cada uno de éstos afecta al la función de coste. Esta derivada, indicará si se debe aumentar o disminuir dicho parámetro en la siguiente iteración para obtener un mejor resultado en la dicha función de coste. Habitualmente, y dado que el número de neuronas de una red neuronal suele ser muy alto, y por lo tanto el número de pesos y sesgos también, se hace computacionalmente poco eficiente utilizar todos los parámetros para la optimización y se utiliza un subconjunto de ellos (minibatches en inglés). Es lo que se llama descenso de gradiente estocástico (SGD por sus siglas en inglés). Una vez determinado si un parámetro debe *moverse* en una dirección u otra, se modifica su valor para la siguiente iteración a través del meta parámetro learning rate, que indica cuánto se debe desplazar cada parámetro en cada iteración.

3.4.7 Learning rate

Como se ha dicho, el learning rate o tasa de aprendizaje indica cuanto se va a modificar cada parámetro en cada iteración para el cálculo del parámetro en la siguiente iteración. Si se establece un parámetro muy pequeño, la red tardará mucho en converger hasta su óptimo y si es muy grande puede pasar por el óptimo sin llegar a alcanzarlo, por eso es uno de los retos más importantes a la hora de diseñar una red neuronal.

Además, un problema que se pueden encontrar las redes neuronales a la hora de calcular los mejores valores para la función objetivo, es la existencia de mínimos locales. A través de la propagación hacia atrás y mediante el descenso de gradiente, la función objetivo irá mejorando entre todos los posibles valores. Puede pasar que la función de coste tenga su valor óptimo en una dirección contraria.

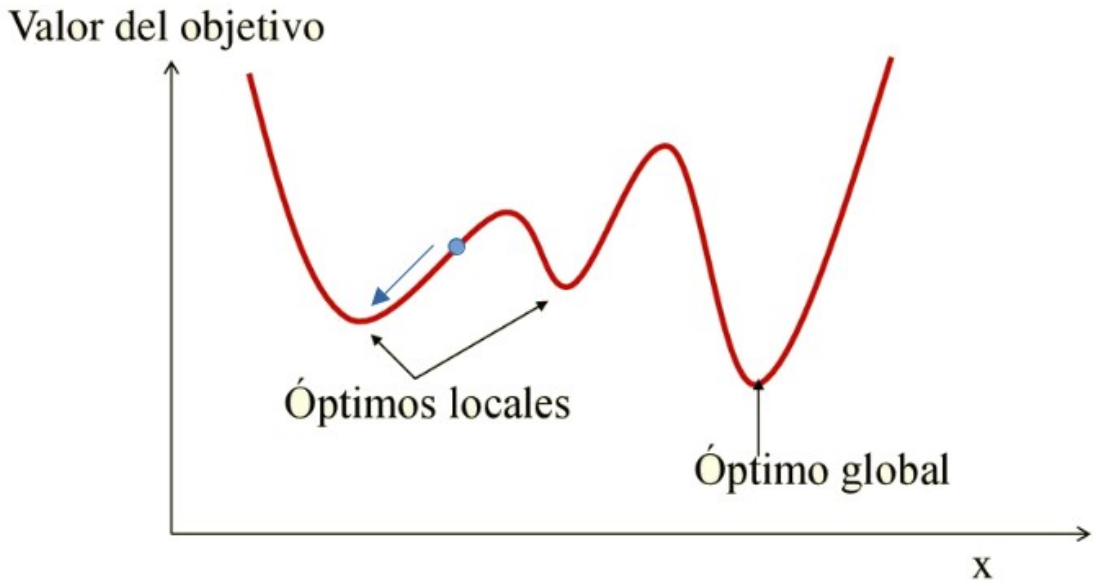

Figura 4: Óptimos locales

Para intentar mejorar mitigar estos problemas se aplican learning rates adaptativos, que modifican dinámicamente el valor del learning rate. Algunos ejemplos de estas técnicas son:

- Momentum (o Stochastic Gracient Descent with Momentum): recuerda el incremento aplicado a las variables en cada iteración y calcula el incremento siguiente

como una combinación lineal entre el gradiente y el incremento anterior, evitando así cambios bruscos en la modificación de la tasa de aprendizaje.

AdaGrad: Modifica las tasas de aprendizaje para cada parámetro individualmente, teniendo en cuenta cuanto se modifica la función de coste con respecto a ese parámetro (a través de su derivada parcial). Puede suceder que la tasa de aprendizaje de un parámetro concreto decrezca demasiado rápidamente porque al principio del entrenamiento la función de coste mejora mucho con la modificación de dicho parámetro, y con ese learning rate pequeño, en iteraciones posteriores le cueste más optimizar.

- RMSProp (Root Mean Square Propagation): Para calcular cómo se modifica el learning rate solo se tienen en cuenta un subconjunto de las últimas mejoras, no la derivada completa, evitando así que la tasa se reduzca demasiado rápidamente.

- Adam (Adaptative Moment Optimización): Aplica la técnica de momentum, pero individualmente para cada parámetro y teniendo en cuenta una ventana de las últimas actualizaciones como RMSProp.

3.5 RED NEURONAL CONVOLUCIONAL

Las redes neuronales convolucionales son un tipo de redes neuronales en las que se tienen en cuenta aspectos espaciales de las entradas de información para extraer características de los mismos. A veces también permite reducir su dimensionalidad (si no se añade padding) y así permite realizar clasificaciones más eficientes. También disponen de mecanismos específicos para reducir la dimensionalidad, como las capas *max pooling*. Estas capas reducen el número de salidas de la capa anterior agrupando los valores de una ventana de esos resultados en uno solo, eligiendo el máximo de esa ventana como se puede ver en la Figura 5.

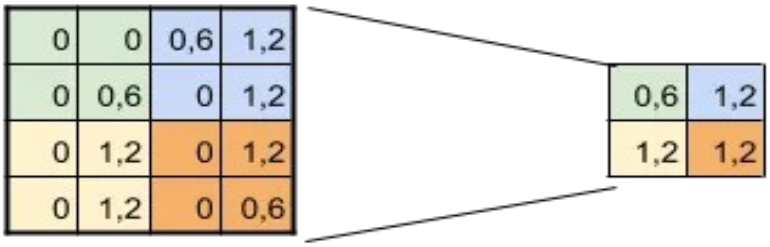

Figura 5: Max pooling - Reducción de características

Los frameworks permite habitualmente combinar redes convolucionales dentro de redes neuronales como cualquier otro tipo de capa, y se puede generar arquitecturas complejas sin necesidad de implementar el paso de parámetros entre unas y otras..

3.5.1 Convolución

La base de las redes convolucionales es la operación de convolución. La convolución consiste en multiplicar una pequeña matriz, que llamaremos kernel o filtro, por cada sección de la entrada. Por ejemplo, podemos aplicar una convolución a una imagen 8x8, con un *kernel* 3x3 para obtener una representación de dicha imagen con una dimensión de 4x4. El *kernel* se va multiplica escalarmente por la parte del mismo tamaño, comenzando por la esquina superior izquierda de la imagen y así se obtiene una salida. Se realiza la tarea tantas veces como es el tamaño de la imagen, deslizando la sección que se multiplica de la imagen hacia la derecha y después de arriba a abajo. El deslizamiento pude ser de uno o más pixels. Así, obtenemos como salida una nueva matriz que representa la presencia o no de las características representados por el *kernel* en cada sección de la imagen.

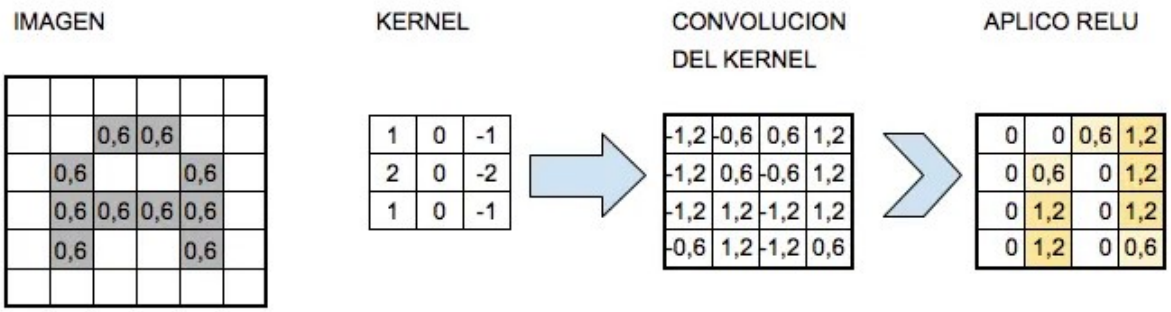

Figura 6: Extracción de características mediante convolución

Existen dos parámetros añadidos en una capa convolucional:

- Padding: es espacio que añade al modelo en sus laterales. El objetivo de usar este parámetro es extraer características de la información que se encuentra en los bordes del modelo, ya que sin el padding, esos pixeles nunca se multiplicarían por los valores centrales del *kernel*.
- Stride: indica cuanto se va a desplazar entre multiplicaciones el *kernel*. Si por ejemplo tiene valor 2, se comienza a multiplicar con la submatriz que comienza en el pixel 0,0, y la siguiente multiplicación será con la que empieza en el 0,2. Lo mismo cuando se va a cambiar de fila, se salta a la 2,0. El valor puede ser diferente en cada una de las dimensiones.

4 EXPERIMENTOS

En este apartado se presentan los experimentos se ha llevado a cabo. Se han comparado los resultados obtenidos con 3 métodos de clasificación sobre la base de datos Modelnet40:

- Red neuronal densa, con la voxelización como entrada.
- Red convolucional 3D, con la voxelización como entrada.
- Red convolucional 2D con los histogramas como entrada.

En cada caso se realizó una clasificación multiclase (40 clases) y además se ha aplicado la técnica one-vs-one para estudiar qué efectos tiene el uso de esta técnica sobre la base de datos, obteniendo como resultado un total de 6 conjuntos de resultados.

A continuación se expondrán el hardware y software utilizados, luego el preprocesado que se ha hecho de los datos (voxelización), después los parámetros generales del entrenamiento y finalmente la arquitectura de las redes aplicadas.

4.1 HARDWARE

Los experimentos y el preprocesado de los datos se han ejecutado en el siguiente Hardware:

- Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz
- 16G RAM
- GeForce 840M 2G
- Driver Version: 440.118.02 CUDA Version: 10.2
- Ubuntu 20.04

4.2 SOFTWARE

Para los experimentos se ha utilizado el software Jupiter[2]. Como software para implementar las distintas configuraciones de redes neuronales se ha utilizado Keras[3]

2 https://jupyter.org/

3 https://keras.io/

(que es una capa de abstracción encima de TensorFlow[4]), que ofrece una forma sencilla y transparente de generar distintas configuraciones de redes, permitiendo al usuario centrarse en la arquitectura y parametrización de la red y no en su implementación. A continuación se puede ver un ejemplo de cómo crear una red y cómo configurarla:

```
model = keras.Sequential([
        keras.layers.Flatten(input_shape=(res, res, res)),
        keras.layers.Dense(res*2, activation='relu'),
        keras.layers.Dense(res/2, activation='sigmoid'),
        keras.layers.Dropout(.2, input_shape=(2,)),
        keras.layers.Dense(num_classes,
activation='softmax')
    ])

model.compile(optimizer=keras.optimizers.Adam(lr=0.0001),
   loss='sparse_categorical_crossentropy',
   metrics=['accuracy'])
```

Código 1: Ejemplo de configuración de red neuronal densa

También permite configurar la fase de entrenamiento para utilizar técnicas como *early stop* y *cross validation* que se explican más adelante.

4.3 PREPROCESADO DE LOS MODELOS: VOXELIZACIÓN

El primer paso para la realización de los experimentos ha consistido en la voxelización de todos los modelos contenidos en la base de datos Modelnet40. El total de modelos contenidos en esta base de datos es de más de 9.800, lo que implica un alto coste de tiempo de computación. La generación de esta voxelización se realizó con ayuda del software opensource *cuda voxelizer*[5], obteniendo como salida, una representación del espacio de 24x24x24. Esta voxelización será la entrada directa de los 4 primeros experimentos realizados (red neuronal densa, red convolucional, y sus correspondientes one vs one) y ha servido como entrada para el cálculo de los histogramas.

4 https://www.tensorflow.org/

5 https://github.com/Forceflow/cuda_voxelizer

4.4 PARÁMETROS DEL ENTRENAMIENTO

El entrenamiento de las distintas arquitecturas de redes neuronales se ha realizado con los mismos parámetros que se detallan a continuación:

4.4.1 Épocas

Aunque en un principio se realizaron experimentos con un número limitado de épocas sobre las que iterar el entrenamiento, posteriormente se incrementó considerablemente ese número (hasta 400), ya que se adoptó la técnica de *early stop* para evitar que el entrenamiento siguiese entrenando sin mejora.

4.4.2 Early Stop[6]

Early stop es una técnica que permite detener el entrenamiento antes de que se completen todas las épocas configuradas. Para ello lo que se hace es indicarle en cuantas épocas anteriores tiene que mirar (parámetro *patient*) y cuanto tiene que mejorar para considerar que el entrenamiento sigue mejorando los parámetros (parámetro *min_delta*). También hay que indicarle sobre qué parámetro se quiere evaluar la mejora (parámetro *monitor*), pudiendo elegir por ejemplo entre el acierto (*accuracy*) o la función de coste (*loss*). Si en las últimas *patient* épocas el parámetro que se está monitorizando no ha mejorado al menos en *min_delta*, el entrenamiento se detendrá.

```
tf.keras.callbacks.EarlyStopping(
                    monitor="val_accuracy",
                    min_delta=0.0001,
                    patience=20,
                    verbose=1,
                    mode="max",
                    baseline=None,
                    restore_best_weights=True,
                )
```

Código 2: Configuración en Keras de la técnica Early Stop

6 https://keras.io/api/callbacks/early_stopping/

4.4.3 Cross validation

La validación cruzada o *cross validation* es una técnica que permite disponer de un valor de acierto en cada época del entrenamiento. Para ello, antes de cada época, se separa un conjunto de instancias que no se utilizan para el entrenamiento, y al final de la época se evalúa la red para esas instancias. En keras este parámetro se llama *validation_split*[7].

4.5 CONFIGURACIÓN DE LAS REDES

A continuación se detallan cómo se han configurado las redes neuronales para los distintos experimentos:

4.5.1 Red Neuronal Densa

Se ha creado mediante Keras una red neuronal con la siguiente configuración:

- Capa *Flatten*: Esta capa "aplana" la entrada tridimensional produciendo un único vector de 13824 posiciones.
- Capa *Dense* (48): Una capa densamente conectada de 48 neuronas y activación *ReLu*.
- Capa *Dense*(12): Una capa densamente conectada de 12 neuronas y activación sigmoide.
- Capa *Dropout*(0.2): Una capa para reducir el sobre entrenamiento, que descarta aleatoriamente.
- Capa *Dense*(40): Una capa de salida con una neurona para cada clase.

4.5.2 Convolución 3D

Este modelo se ha compuesto de la siguiente configuración en Keras:

- *Conv3D*[8], con un *kernel* de tamaño 3x3x3 y un salto entre voxels de 2. La función de activación utilizada ha sido *ReLu*.

7 https://keras.io/api/models/model_training_apis/

8 https://keras.io/api/layers/convolution_layers/convolution3d/

- *Conv3D*, con un kernel de tamaño 3 y un salto de 3. Con esta capa conseguimos reducir aún más la dimensionalidad del problema.
- *Flaten*[9]: Con esta capa se aplana la salida para poder seguir utilizándola en la red.
- *Dense*[10](24): Se crea una capa densa para mejorar el entrenamiento.
- *Dense*(40): Se añade una capa de salida.

Este modelo está inspirado en la configuración utilizada en el trabajo original de ModelNet que tenía la siguiente configuración:

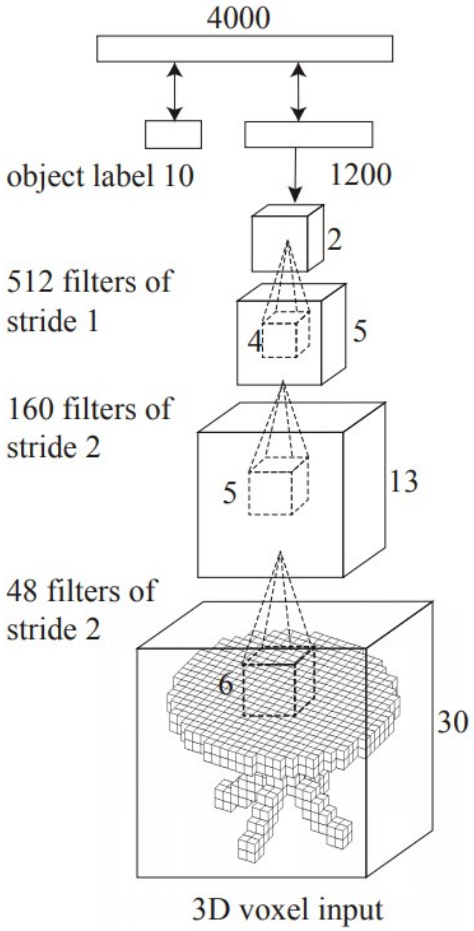

Figura 7: Arquitectura de Modelnet

9 https://keras.io/api/layers/reshaping_layers/flatten/

10 https://keras.io/api/layers/core_layers/dense/

4.5.3 Histogramas

Para este experimento se ha calculado el histograma de ocupación del espacio en cada dimensión para cada uno de los modelos de los conjuntos de entrenamiento y test. Posteriormente, tomándolos como entrada, se ha entrenado una red neuronal con las siguientes capas:

- Conv2D[11]: Una capa de convolución, en este caso 2D, para extraer las características de la representación del histograma, que tiene una dimensionalidad inicial de 24x3. Se le aplica un kernel de 2x3 sin salto.
- MaxPooling 2D[12]: una capa de pooling para abstraer las características extraídas de su posición espacial.
- Flatten: una capa para conectar con el resto de la red.
- Dense (12): Una capa densa para mejorar el entrenamiento.
- Dense (40): Una capa de salida con cada clase de modelo a clasificar.

4.5.4 Red neuronal densa + One vs. One

La red neuronal utilizada es similar es similar a la utilizada en el punto 4.5.1, con la excepción de la capa de salida que es binaria.

4.5.5 Conv3D + One vs. One

Al igual que en el caso anterior, se aplicó la misma red convolucional que en 4.5.2 a cada pareja de clases, con la excepción del la capa de salida que era binaria.

4.5.6 Histogramas + ONE VS. ONE

En este experimento se repitió el proceso de los anteriores: utilizando una red neuronal similar a la del punto 4.5.3, con la excepción de la capa de salida que en este caso era binaria.

11 https://keras.io/api/layers/convolution_layers/convolution2d/

12 https://keras.io/api/layers/pooling_layers/max_pooling2d/

4.6 RESULTADO OBTENIDOS

En la siguiente tabla se comparan los distintos métodos con su correspondiente aplicación de la metodología one vs. one, indicando la mejoría o empeoramiento que produce dicha aplicación:

Tabla 1: Comparación metodologías

Método	Multiclase	+ One vs. One	% Mejora
Voxelización RN	0,6487	0,7092	9,33
Voxelización Conv3D	0,7220	0,75	3,87
Histograma	0,7331	0,5181	-29,32

Analizando estos resultados se pueden extraer las siguientes conclusiones:

- La clasificación mediante histogramas obtiene la mejor tasa de acierto sin aplicar la binarización, a pesar de que se reduce considerablemente el tamaño de los datos de entrada para la red neuronal creada. Esto es beneficioso para el modelo, ya que le permite realizar la clasificación de una nueva instancia en menor tiempo, pero hay que tener en cuenta que requiere un preprocesado.
- La binarización one vs one mejora los resultados de la voxelización, que es lo que habríamos esperado del experimento. Es especialmente notoria la mejora obtenida en el caso de la red neuronal densa, y puede ser debido a que el método original tenía una tasa de acierto bastante baja, con lo que su capacidad de mejora era mayor que en el caso de la convolución.
- La binarización empeora el resultado de la clasificación de por histogramas. No está claro el porqué de este resultado no esperado. Una observación de la Tabla 2 indica que en muchas clases la clasificación empeora considerablemente, pero no se ha conseguido extraer conclusiones de los motivos. Incluso durante la ejecución del entrenamiento de los clasificadores por parejas individuales se observa que hay parejas sobre las que parece muy difícil actuar a la red neuronal. Esto puede significar que la pérdida de

información que se produce con la extracción de los histogramas es perjudicial.

- Por último se puede ver que el experimento que más se asemeja al trabajo original de Modelnet (Voxelización con convolución 3D) aún se queda algo alejado de los resultados del original (0,77). Esto puede ser debido a diferencias en el tratamiento de los datos originales (diferente particionado del espacio de voxelización, adición de paddings…) o hiperparámetros utilizados en la configuración de las capas aplicadas.

4.6.1 Comparación por clase

En la siguiente tabla se comparan los distintos métodos según su tasa de acierto en cada una de las 40 clases en las que se dividían los modelos:

Tabla 2: Comparación de métodos por clase

Clase	Dense	Conv3D	Dense + 1vs1	Conv3D + 1vs1	Histo	Histo + 1vs1	Instancias
airplane	0.81	0.93	0.80	0.91	0.86	0.54	626
bathtub	0.82	0.88	0.92	0.97	0.82	0.71	106
bed	0.63	0.82	0.76	0.76	0.79	0.47	515
bench	0.38	0.45	0.48	0.54	0.47	0.16	173
bookshelf	0.53	0.73	0.59	0.66	0.78	0.41	572
bottle	0.65	0.74	0.73	0.81	0.72	0.62	335
bowl	0.60	0.61	0.62	0.75	0.75	1.00	64
car	0.85	0.90	0.83	0.91	0.93	0.86	197
chair	0.70	0.84	0.60	0.77	0.75	0.27	889
cone	0.53	0.60	0.58	0.79	0.65	1.00	167
cup	0.00	0.38	0.50	0.19	0.12	0.00	79
curtain	0.50	0.48	0.45	0.52	0.44	0.29	138
desk	0.53	0.63	0.58	0.62	0.68	0.51	200
door	0.60	0.55	0.61	0.54	0.59	0.64	109
dresser	0.66	0.59	0.70	0.74	0.65	0.55	200

flower_pot	0.00	0.03	0.05	0.04	0.12	0.00	149
glass_box	0.75	0.86	0.83	0.91	0.84	0.91	171
guitar	0.99	0.96	0.96	0.99	0.96	0.61	155
keyboard	0.46	0.33	0.56	0.53	0.59	0.69	145
lamp	0.50	0.48	0.42	0.50	0.56	0.56	124
laptop	0.58	0.77	0.70	0.58	0.57	0.83	149
mantel	0.80	0.87	0.93	0.92	0.90	0.70	284
monitor	0.79	0.86	0.83	0.84	0.88	0.60	465
night_stand	0.57	0.66	0.66	0.64	0.69	0.52	200
person	0.00	0.38	0.58	0.46	0.50	0.33	88
piano	0.73	0.75	0.73	0.73	0.73	0.41	231
plant	0.50	0.57	0.60	0.61	0.78	0.61	240
radio	0.00	0.25	0.33	0.14	0.27	0.00	104
range_hood	0.93	0.95	0.99	0.94	0.93	1.00	115
sink	0.29	0.33	0.55	0.55	0.16	0.00	128
sofa	0.76	0.90	0.90	0.92	0.90	0.71	680
stairs	0.00	0.44	0.75	0.78	0.38	0.00	124
stool	0.00	0.62	0.58	0.50	0.48	1.00	90
table	0.70	0.81	0.68	0.68	0.78	0.66	392
tent	0.24	0.36	0.41	0.50	0.37	0.33	163
toilet	0.92	0.88	0.97	0.96	0.94	0.90	344
tv_stand	0.42	0.58	0.65	0.72	0.64	0.39	267
vase	0.32	0.60	0.39	0.57	0.52	0.38	475
wardrobe	0.00	0.27	0.80	0.67	0.31	0.50	87
xbox	0.00	0.50	1.00	0.89	0.33	0.67	103

De estos datos se pueden extraer algunas conclusiones:

- Algunas clases se identifican muy mal con todos los clasificadores. Varias de ellas pueden deberse a que cuentan con menos instancias en el conjunto de entrenamiento como se ve en la Tabla 2, pero sería interesante realizar un

estudio pormenorizado al respecto ya que hay categorías que aún con un número bajo de instancias clasifican muy bien (por ejemplo las campanas extractoras, *range_hood*).

- En los casos de voxelización, aunque por regla general la binarización mejora los resultados, hay algunas clases en las que no es así. Habría que recurrir a un análisis profundo de los resultados parciales de los clasificadores de la binarización para determinar si esas clases tienen problemas generales en la clasificación o por el contrario únicamente cuando se enfrentan a otras clases concretas.
- Como se ha observado en los resultados generales, la binarización para el análisis de los histogramas ofrece datos peores para casi todas las clases por lo se puede deducir que no es que un conjunto de clases con resultados malos lastre la clasificación general.

5 CONCLUSIONES Y FUTUROS TRABAJOS

La identificación de objetos 3D es un área del aprendizaje automático donde aún queda mucho margen de mejora, sobre todo en lo referente al tratamiento de los modelos recibidos. La aplicación de nuevas técnicas de extracción de características como las convoluciones 3D, o enfoques de división del problema en problemas menos complejos, como la binarización ofrecen, generalmente, una mejora en los resultados.

Se ha observado que hay una serie de clases que clasifican peor que otras, y se ha encontrado que se corresponden con que son clases que tienen menos modelos para realizar el entrenamiento. Sería interesante ampliar el número de modelos de estas clases para ver si con ello mejora su clasificación, por ejemplo, aplicando transformaciones espaciales a los modelos disponibles (rotaciones, inversiones…) o con la búsqueda de nuevos modelos a tratar. En futuros trabajos sería interesante estudiar si los filtros de la convolución pueden ser capaces de extraer características independientemente, no ya de la posición espacial del modelo, si no también de su orientación, con lo que haría innecesaria esta propuesta.

También se ha observado que para esta clasificación no se han tenido en cuenta las verdaderas dimensiones del modelo, ocupando el mismo espacio un avión que una botella. Este podía ser una entrada muy importante a la hora de realizar una clasificación más exacta y debería ser tenida en cuenta en los futuros trabajos.

También resultaría interesante realizar un análisis de combinación de los distintos clasificadores utilizados para comprobar en qué medida las carencias de uno de los métodos pueden ser suplidas por el resto.

6 BIBLIOGRAFÍA

[1] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao, 3D ShapeNets: A Deep Representation for Volumetric Shapes Proceedings of 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR2015) Oral Presentation · 3D Deep Learning Project Webpage

[2] Zizhao Zhang, Haojie Lin, Xibin Zhao, Rongrong Ji, Senior Member, IEEE, Yue Gao, Senior Member, IEEE, Inductive Multi-Hypergraph Learning and Its Application on View-Based 3D Object Classification

[3] Asako Kanezaki, Yasuyuki Matsushita, Yoshifumi Nishida, RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints

[4] Charles R. Qi, Hao Su, Kaichun Mo, Leonidas J. Guibas, PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

[5] N. Sedaghat and M. Zolfaghari and E. Amiri and T. Brox, Orientation-boosted voxel nets for 3D object recognition

[6] Chinnu A , MRI Brain Tumor Classification Using SVM and Histogram Based Image Segmentation

[7] E. Cheng, N. Xie, H. Ling, P. R. Bakic, A. D. A. Maidment and V. Megalooikonomou, "Mammographic image classification using histogram intersection," 2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Rotterdam, 2010, pp. 197-200, doi: 10.1109/ISBI.2010.5490381.

[8] Bishop, Christopher, Pattern Recognition and Machine Learning

[9] Park SH., Fürnkranz J. (2007) Efficient Pairwise Classification. In: Kok J.N., Koronacki J., Mantaras R.L.., Matwin S., Mladenič D., Skowron A. (eds) Machine Learning: ECML 2007. ECML 2007. Lecture Notes in Computer Science, vol 4701. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-74958-5_65

[10] Nguyen N., Caruana R. (2008) Improving Classification with Pairwise Constraints: A Margin-Based Approach. In: Daelemans W., Goethals B.,

Morik K. (eds) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2008. Lecture Notes in Computer Science, vol 5212. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-87481-2$_8$