PointNet: A 3D Convolutional Neural Network for Real-Time Object Class Recognition

A. Garcia-Garcia*, F. Gomez-Donoso†, J. Garcia-Rodriguez*, S. Orts-Escolano*, M. Cazorla†, J. Azorin-Lopez*
*Department of Computer Technology, University of Alicante
{agarcia, jgarcia, sorts, jazorin}@dtic.ua.es
†Department of Computer Science and Artificial Intelligence, University of Alicante
fgomez@dccia.ua.es, miguel.cazorla@ua.es

Abstract—**During the last few years, Convolutional Neural Networks are slowly but surely becoming the default method to solve many computer vision related problems. This is mainly due to the continuous success that they have achieved when applied to certain tasks such as image, speech, or object recognition.**

Despite all the efforts, object class recognition methods based on deep learning techniques still have room for improvement. Most of the current approaches do not fully exploit 3D information, which has been proven to effectively improve the performance of other traditional object recognition methods.

In this work, we propose *PointNet*, a new approach inspired by *VoxNet* and *3D ShapeNets*, as an improvement over the existing methods by using density occupancy grids representations for the input data, and integrating them into a supervised Convolutional Neural Network architecture.

An extensive experimentation was carried out, using *ModelNet* – a large-scale 3D CAD models dataset – to train and test the system, to prove that our approach is on par with state-of-the-art methods in terms of accuracy while being able to perform recognition under real-time constraints.

I. Introduction

Object class recognition is one of the main challenges for a computer to be able to understand a scene. This turns out to be a key capability for autonomous robots which operate in real-word environments. Due to the unstructured nature of those environments, mobile robots need to do reasoning grounded in the real world, i.e., they must understand the information provided by their sensors. In this regard, semantic object recognition is a crucial task, among other equally important ones, which robots have to perform in order to achieve a considerable level of scene understanding [1].

The advent of reliable and low-cost range sensors, e.g., RGB-D cameras such as Microsoft Kinect and PrimeSense Carmine, revolutionized the field by providing useful 3D data to feed the prediction systems with a new dimension of useful information. Because of that, traditional 2D approaches were superseded by 3D-based ones. In addition, 3D model and scene repositories are growing on a daily basis, thus providing researchers with enough reliable data sources for training and testing object recognition systems.

The vast majority of 3D object recognition methods [2] are typically based on hand-crafted local feature descriptors [3]. These kinds of approaches rely on specific pipelines [4] consisting of a keypoint detection phase, followed by the computation of descriptors at those characteristic regions, finally they are classified to determine the possible object represented by those descriptors. That classification is performed by using distance metrics or machine learning algorithms, e.g., Support Vector Machines (SVMs) [5], random forests [6], or neural networks, which are trained with object datasets. Despite the popularity and the continuous flow of successful implementations of this kind of pipelines – specially for recognition in cluttered and occluded environments – hand-crafting feature descriptors requires domain expertise and remarkable engineering and theoretical skills, and even fulfilling both requirements they are still far from perfection. In this regard, the aim of researchers has been to replace those hand-engineered descriptors with multilayer neural networks able to learn them automatically by using general-purpose training algorithms. This gave birth to the application of Convolutional Neural Networks (CNNs) to image analysis. A brand-new deep learning architecture designed to process data in form of multiple arrays, which quickly surpassed the previous methods with many practical successes [7] – mainly due to the fact that they were easy to train and generalized far better – so that CNNs have become the community standard to tackle the object class recognition problem [8]. However, only a few CNN-based systems have been proposed to use 3D data for this purpose, therefore we strongly believe that there is still room for improvement since most of them do not fully exploit 3D data.

In this work, we propose a new deep learning architecture for object class recognition using pure 3D information, an approach inspired by the success of recently introduced CNN-based systems for 3D object recognition. Its contribution is twofold: a novel way for representing the input data, which is based on point density occupancy grids, and its integration into a CNN specifically tuned for the aforementioned purpose.

This paper is structured as follows. Section II reviews the state of the art of deep learning methods applied to 3D object class recognition. Next, our proposal – namely *PointNet* – is described in Section III. Section IV defines the methodology followed to test our approach, as well as the experiments that were carried out, their results, and the corresponding discussion. At last, Section V draws some conclusions about our work and sets future research lines to improve the proposal.

978-1-5090-0620-5/16/$31.00 ©2016 IEEE

II. Related Work

Deep learning architectures have recently revolutionized 2D object class recognition. The most significant example of such success is the CNN architecture, being *AlexNet* [7] the milestone which started that revolution. Krizhevsky *et al.* developed a deep learning model based on the CNN architecture that outperformed by a large margin ($15.315\ \%$ error rate against the $26.172\ \%$ scored by the runner-up not based on deep learning) state-of-the-art methods on the *ImageNet* [9] LSVRC 2012 challenge.

Due to the successful applications of the CNNs to 2D image analysis, several researchers decided to take the same approach for 2.5D data, treating the depth channel as an additional one together with the RGB ones [10]–[12]. These methods simply extend the architecture to take four channels – matrices – as input instead of the three featured by RGB images. This is still a image-based approach which does not fully exploit the geometric information of 3D shapes despite its straightforward implementation. Apart from 2.5D approaches, specific architectures to learn from volumetric data, which make use of pure 3D convolutions, have been recently developed. Those architectures are commonly referred as 3DCNNs and their foundations are the same as the 2D or 2.5D ones, but the nature of the input data is radically different. Since volumetric data is usually quite dense and hard to process, most of the successful 3DCNNs resort to a more compact representation of the 3D space: the occupancy grids. *VoxNet* [13] and *3D ShapeNets* [14] make extensive use of this representation.

Those 3DCNNs are slowly overtaking other approaches when applying object recognition to complete 3D scenes [15]. This progress has been mainly enabled by two factors: the substantial growth in the number of 3D models available online through repositories, and the reduction of training times thanks to frameworks and libraries which exploit the power of massively parallel architectures for this kind of tasks. On the one hand, there exist many collections of 3D models, but they tend to be small and usually lack annotations and other useful information for training this kind of deep architectures. In contrast, 2D approaches have taken advantage of the numerous and high-quality datasets that already exist such as *ImageNet* [9], *LabelMe* [16], and *SUN* [17]. During the last years, researchers have unified efforts to create large-scale annotated 3D datasets inspired by the success of the 2D counterparts. The most popular 3D datasets which have revamped data-driven solutions – for computer vision in general, and object recognition in particular – are the *Princeton ModelNet* [14], and *ShapeNets* [18] datasets. On the other hand, the creation of deep learning frameworks such as *Caffe* [19], *Theano* [20], *Torch* [21], or *TensorFlow* [22], which allow researchers to easily express and launch their architectures and accelerate the training calculations with Graphics Processing Units (GPUs) by using CUDA or OpenCL, has enabled quick prototyping and testing. Both facts have turned out to be crucial for the development of the field.

III. Approach

The proposed system takes a point cloud of an object as an input and predicts its class label. In this regard, the proposal is twofold: a volumetric grid based on point density to estimate spatial occupancy inside each voxel, and a pure 3DCNN which is trained to predict object classes. The occupancy grid – inspired by *VoxNet* [13] occupancy models based on probabilistic estimates – provides a compact representation of the object's 3D information from the point cloud. That grid is fed to the CNN architecture, which in turn computes a label for that sample, i.e., predicts the class of the object.

This architecture was implemented using the Point Cloud Library (PCL) [23] – which contains state-of-the-art algorithms for 3D point cloud processing – and *Caffe* [19], a deep learning framework developed and maintained by the Berkeley Vision and Learning Center (BVLC) and an active community of contributors on *GitHub*[1]. This BSD-licensed C++ library allows us to design, train, and deploy CNN architectures efficiently, mainly thanks to its drop-in integration of NVIDIA *cuDNN* [24] to take advantage of GPU acceleration.

A. Occupancy Grid

Occupancy grids [25] are data structures which allow us to obtain a compact representation of the volumetric space. They stand between meshes or clouds, which offer rich but large amounts of information, and voxelized representations with packed but poor information. At that midpoint, occupancy grids provide considerable shape cues to perform learning, while enabling an efficient processing of that information thanks to their array-like implementation.

Recent 3D deep learning architectures make use of occupancy grids as a representation for the input data to be learned or classified. *3D ShapeNets* [14] is a Convolutional Deep Belief Network (CDBN) which represents a 3D shape

[1] http://github.com/BVLC/caffe

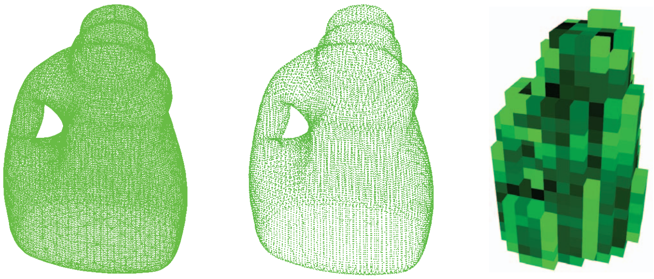

(a) Mesh (b) Cloud (c) Occ. Grid

Fig. 1: A mesh (a) is transformed into a point cloud (b), and that cloud is processed to obtain a voxelized occupancy grid (c). The occupancy grid shown in this figure is a cube of $30 \times 30 \times 30$ voxels. Each voxel of that cube holds the point density inside its volume. In this case, dark voxels indicate high density whilst bright ones are low density volumes. Empty voxels were removed for better visualization.

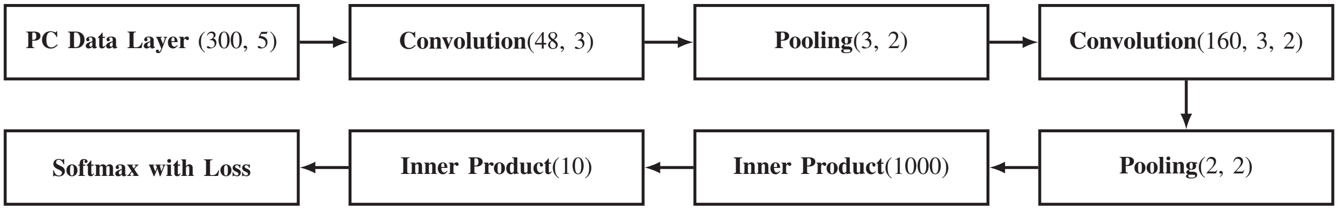


Fig. 2: *PointNet* Convolutional Neural Network architecture.

as a 30 × 30 × 30 binary tensor in which a one indicates that a voxel intersects the mesh surface, and a zero represents empty space. *VoxNet* [13] introduces three different occupancy grids (32 × 32 × 32 voxels) that employ 3D ray tracing to compute the number of beams hitting or passing each voxel and then use that information to compute the value of each voxel depending on the chosen model: a binary occupancy grid using probabilistic estimates, a density grid in which each voxel holds a value corresponding to the probability that it will block a sensor beam, and a hit grid that only considers hits thus ignoring empty or unknown space. The binary and density grids proposed by Maturana *et al.* differentiate unknown and empty space, whilst the hit grid and the binary tensor do not.

Currently, *VoxNet*'s occupancy grid holds the best accuracy in the *ModelNet* challenge for the 3D-centric approaches described above. However, ray tracing grids considerably harmed performance in terms of execution time so that other approaches must be considered for a real-time implementation. In that very same work, the authors show that hit grids performed comparably to other approaches while keeping a low complexity to achieve a reduced runtime.

In this regard, we propose an occupancy grid inspired by the aforementioned successes but aiming to maintain a reasonable accuracy while allowing a real-time implementation. In our volumetric representation, each point of a cloud is mapped to a voxel of a fixed-size occupancy grid. Before performing that mapping, the object cloud is scaled to fit the grid. Each voxel will hold a value representing the number of points mapped to itself. At last, the values held by each cell are normalized. Figure 1 shows the proposed occupancy grid representation for a sample object.

B. Network Architecture

As we have previously stated, CNNs have proven to be very useful for recognizing and classifying objects in 2D images. A convolutional layer can recognize basic patterns such as corners or planes and if we stack several of them they can learn a topology of hierarchical filters that highlight regions of the images. What is more, the composition of several of these regions can define a feature of a more complex object. In this regard, a combination of various filters is able to recognize a full object. We apply this approach used in 2D images to 3D recognition. In this section we will describe the main layers that compose *PointNet* as well as their parameters.

Point Cloud Data Layer PC(v, l): This layer takes as input a point cloud and outputs an occupancy grid structure. The tasks performed by this layer are defined in Section III-A. This layer takes the following parameters:

- Voxel Grid Size (v): It defines the width, height, and depth of the occupancy grid in spatial units. A value of 300 is used to carry out the experiments in this paper.
- Leaf Size (l): It specifies the width, height and depth of a single voxel in spatial units. A value of 5 is used for our architecture.

In sight of these parameters, an occupancy grid data structure of $60 \times 60 \times 60$ voxels is used to process the input point clouds.

Convolution Layer C(m, n, d) : These layers convolve the data in batches of $n \times n \times n$ cubes with m filters and a stride of d. As a result, they provide m grids that are the response of each filter. The values of the filters are updated as they are learned in the backpropagation stage. They are initialized with random values at the beginning. The hyperparameters are:

- Number of filters (m): It defines how many filters the layer will learn.
- Filter size (n): If specifies the width, height and depth of the filter.
- Stride (d): This parameter sets the offset to apply during the convolution process.

ReLU Layer R(): This layer performs the operation $max(x, 0)$ returning the same value if the input value is greater or equal to zero, and zero if the input value is negative. This layer takes no parameters. The first *Inner Product* layer is followed by a *ReLU* one together with a dropout layer to avoid overfitting with a 0.5 ratio.

Pooling Layer P(n,d): In this layer, a max-pooling process is performed. It takes the input data and summarizes it by taking the max value of a fixed local spatial region sliding it across the grid data structure.

- Filter size (n): It specifies the width, height and depth of the local spatial regions.
- Stride (d): This parameter sets the offset to apply in the pooling process.

Inner Product Layer IP(n): This is a fully connected layer in which every input neuron is connected to each output one throughout a hyperparameter. These hyperparameters are learned during the backpropagation process.

- Number of outputs (n): It determines the number of output neurons.

The deep architecture featured by *PointNet* is represented in Figure 2. This setup allows *PointNet* to be on par with state-of-the-art algorithms while keeping reduced execution times.

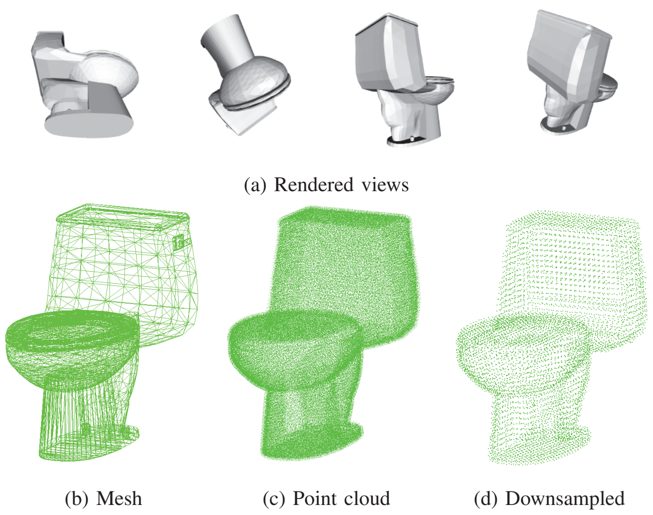
IV. Experiments

In order to evaluate the performance of our proposal in terms of accuracy we made extensive use of a well-known dataset such as the *Princeton ModelNet* project [14]. Its goal, as their authors state, is to provide researchers with a comprehensive clean collection of 3D CAD models for objects, which were obtained via online search engines. Employees from the Amazon Mechanical Turk service were hired to classify over 150000 models into 662 categories.

At the moment, there are two versions of this dataset publicly available for download [2]: *ModelNet-10* and *ModelNet-40*. Those are subsets of the original dataset, only providing the 10 and 40 most popular object categories respectively. They are specially clean since the models that did not belong to the specified categories were manually deleted.

On the one hand, *ModelNet-10* is composed of a collection of over 5000 CAD models classified into 10 categories and divided into training and test sets. In addition, the orientation of all the CAD models was manually aligned. On the other hand *ModelNet-40* features over 9800 models classified into 40 categories and it also includes training and test splits; however, their orientations are not aligned as they are in *ModelNet-10*.

Table I shows the model distribution for both *ModelNet-10* and *ModelNet-40* per each object category and also the training and test splits.



(a) Rendered views

(b) Mesh (c) Point cloud (d) Downsampled

Fig. 3: Dataset model processing example to generate the point clouds for *PointNet*. Some rendered views of a toilet model are shown in (a). The original OFF mesh is shown in (b). The generated point cloud after merging all points of view is shown in (c), and (d) shows the downsampled cloud using a voxel grid filter with a leaf size of $0.7 \times 0.7 \times 0.7$.

	ModelNet-10		*ModelNet-40*	
Category	**Training Set**	**Test set**	**Training Set**	**Test set**
Desk	200	86	200	86
Table	392	100	392	100
Nighstand	200	86	200	86
Bed	515	100	515	100
Toilet	344	100	344	100
Dresser	200	86	200	86
Bathtub	106	50	106	50
Sofa	680	100	680	100
Monitor	465	100	465	100
Chair	889	100	889	100
Airplane	-	-	626	100
Bench	-	-	173	20
Bookshelf	-	-	572	100
Bottle	-	-	335	100
Bowl	-	-	64	20
Car	-	-	197	100
Cone	-	-	167	20
Cup	-	-	79	20
Curtain	-	-	138	20
Door	-	-	109	20
Flower Pot	-	-	149	20
Glass Box	-	-	171	100
Guitar	-	-	155	100
Keyboard	-	-	145	20
Lamp	-	-	124	20
Laptop	-	-	149	20
Mantel	-	-	284	100
Person	-	-	88	20
Piano	-	-	231	100
Plant	-	-	240	100
Radio	-	-	104	20
Range Hood	-	-	115	100
Sink	-	-	128	20
Stairs	-	-	124	20
Stool	-	-	90	20
Tent	-	-	163	20
TV Stand	-	-	267	100
Vase	-	-	475	100
Wardrobe	-	-	87	20
X-Box	-	-	103	20

TABLE I: ModelNet-10 and ModelNet-40 distributions.

A. Methodology

The CAD models are provided in Object File Format (OFF). Firstly, we converted all OFF models into Polygon File Format (PLY) to ease the usage of the dataset with the PCL. As we already mentioned, the input for *PointNet* are point clouds, but the dataset provides CAD models specifying vertices and faces. In this regard, we converted the PLY models into Point Cloud Data (PCD) clouds by raytracing them. A 3D sphere is tessellated and a virtual camera is placed in each vertex of that truncated icosahedron – pointing to the origin of the model – then multiple snapshots are rendered using raytracing and the z-buffer data, which contains the depth information, is used to generate point clouds from each point of view. After all points of view have been processed, the point clouds are merged. A voxel grid filter is applied to downsample the clouds after the raytracing operations. Figure 3 illustrates the aforementioned processes. After that, the resulting point clouds are used to train, randomizing the order of the models, and test the system taking into account the corresponding splits.

[2] http://modelnet.cs.princeton.edu/

Class	Desk	Table	Nstand	Bed	Toilet	Dresser	Bathtub	Sofa	Monitor	Chair
Desk	52	9	1	4	0	5	1	5	0	9
Table	25	69	0	1	0	0	0	0	1	4
Nstand	1	2	60	1	4	8	0	0	2	8
Bed	4	0	0	80	0	0	3	11	1	1
Toilet	1	0	3	1	84	0	1	3	2	5
Dresser	3	0	14	0	0	61	0	1	6	1
Bathtub	0	1	0	3	0	0	34	8	3	1
Sofa	1	0	1	4	1	2	0	88	1	2
Monitor	1	1	1	1	0	5	1	1	87	2
Chair	1	2	1	2	1	1	0	1	1	90

Fig. 4: Confusion matrix of the classification results achieved by *PointNet* after 200 training iterations using the *ModelNet-10* dataset (learning rate is 0.0001, momentum is 0.9). It is important to remark the confusion between the classes Desk and Table. The values shown in the table are not percentages but absolute number of examples.

All the timings and results were obtained by performing the experiments in the following test setup: Intel Core i5-3570 with 8 GB of 1600 MHz DD3 RAM on an ASUS P8H77-M PRO motherboard (Intel H77 chipset). Additionally, the system includes an NVIDIA Tesla K20 GPU, and a Seagate Barracuda 7200.14 secondary storage.

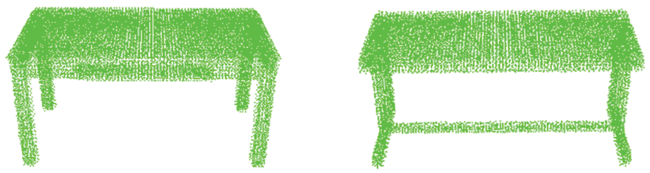
Caffe RC2 was run over ElementaryOS Freya 0.3.1, an Ubuntu-based Linux distribution. It was compiled using CMake 2.8.7, g++ 4.8.2, CUDA 7.0, and cuDNN v3.

B. Results

As a result of training *PointNet* with a learning rate of 0.0001 and a momentum of 0.9 during 200 iterations using the *ModelNet-10* dataset, it obtained a success rate of 77.6%.

As shown in Figure 4, the confusion matrix reveals the stability of the system, mainly confusing items that look alike such as desk and table. Because of the nature of the CNNs, which heavily rely on detecting combinations of features, these kind of errors are common. As we can observe in Figure 5, the visual features that define a desk and a table are almost the same, making it hard to distinguish. Figure 6 shows the neuron activations for the output layer of the architecture, proving that Desk and Table are consistently confused during the tests.

In light of these experiments, and taking into accout the knowledge of the CNNs principles, it is conceivable to think that a deeper network would provide better results so more experiments were carried out.


(a) Table

(b) Desk

Fig. 5: Similarity between two objects of different classes: Table and Desk. The point cloud shown in (a) represents an object of the Table class, whilst the point cloud in (b) represents an object whose class is Desk but it is misclassified as a Table due to their resemblance.

In the deeper network experiment we added several layers to the *PointNet* architecture. One more convolutional layer was added since these layers are coupled to the detection of the features of the objects, so the more layers there are, a better or more expressive model is produced. An Inner Product layer was also added. Since these layers make the classification possible, adding more of them would theoretically provide better classification results. The proposed architecture for this experiment is described as follows: *PC(300,5) - C(64,3) - R() - P(3,2) - C(160,3,2) - R ()- P(2,2) - C(160,3,2) - R() - P(2,2) - IP(1000) - IP(1000) - IP(10)*.

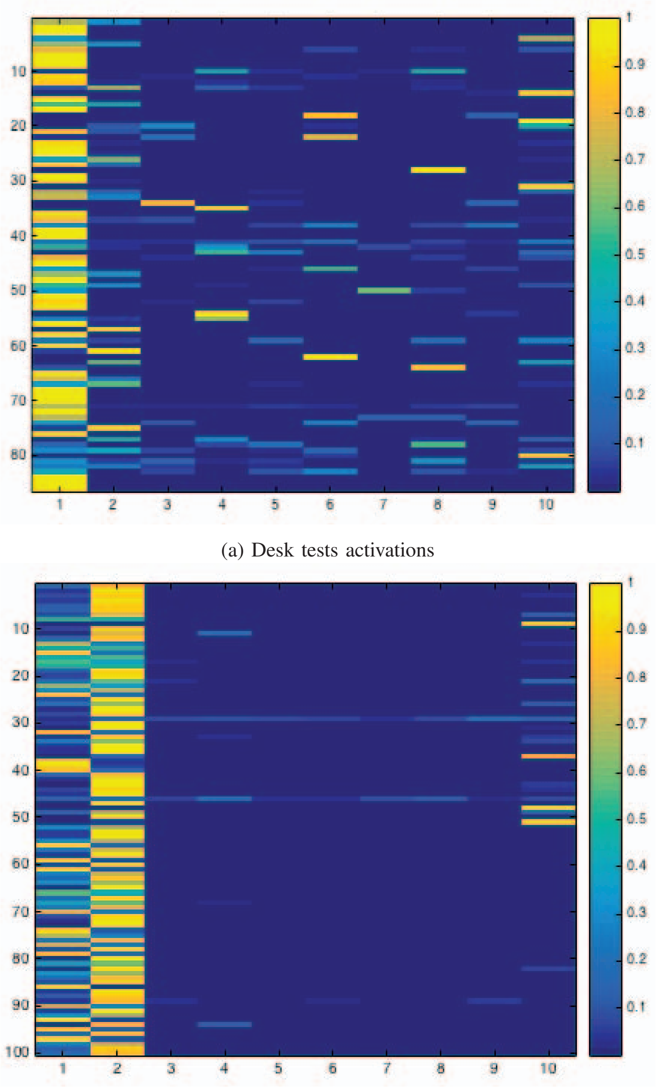
This architecture was trained during 1,000 iterations and tested every 200 iterations. The best result was provided by the 800 iterations test with an accuracy of 76.7%, while the 1,000 iterations test dropped the performance to a 75.9% due to overfitting.

It is well known that training using an unlabanced dataset tends to harm those classes with the least number of examples and to benefit those with the most, as stated by [26]. Having this in mind, and knowing that *ModelNet-10* is highly unbalanced as shown in Table I, the dataset was balanced by limiting the number of examples of each class to 400 using random undersampling. This does not fully solve the problem but improves the difference between the classes with the least number of examples and those with the most.

The network was trained and tested with this more balanced dataset using the architecture defined in Section III-B and it achieved an accuracy of 72.9%. The fact is that balancing the training set makes the accuracy of the classes with less examples higher, but it harms the success rate on classes with more, as seen in Figure 7.

C. Discussion

After analyzing the results, it can be stated that neither a deeper network nor balancing the dataset increase accuracy. In fact, the experiments of the original architecture with the unbalanced *ModelNet-10* offered the best recognition results



(a) Desk tests activations

(b) Table tests activations

Fig. 6: Neuron activations for the output layer of the architecture when classifying all the test samples for both Desk and Table classes. Each row represents an activation vector for a specific sample, so each column is a position of the vector: the activation to that particular class. The first column corresponds to the Desk class, while the second one is the Table. The activation shows the clear confusion between Desk and Table. Although the latter one is much less confused with other classes, many Tables are misclassified as Desks thus lowering the accuracy for this class.

with a 77.6% success rate. In addition, *PointNet* featuring the architecture exposed in Figure 2 takes an average time of 24.6 miliseconds to classify an example (in comparison with *Voxnet*, which can take up to half a second for its raytracing-based implementation). These results prove the system as a fast and accurate 3D object class recognition tool.

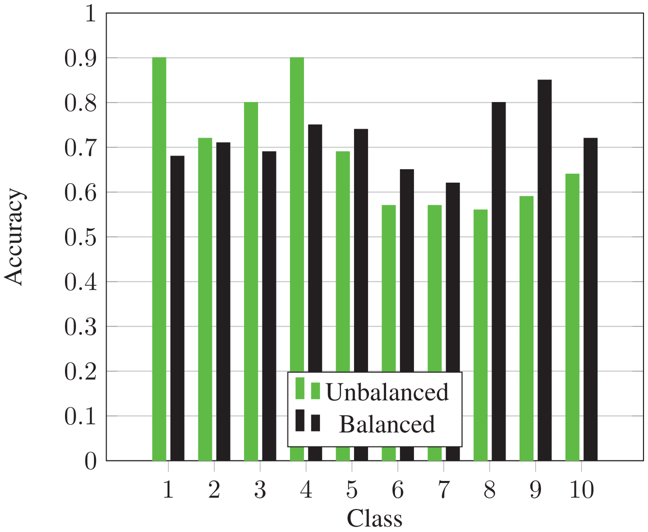


Fig. 7: Comparison of accuracy per class using an unbalanced dataset and a balanced one with a maximum of 400 models per class. Accuracy is lost in the classes in which models are removed but gained otherwise.

V. Conclusion

In this paper we proposed *PointNet*, a brand new kind of CNN for object class recognition that handles tridimensional data, inspired by *VoxNet* and *3D ShapeNets* but using density occupancy grids as inner representation for input data. It was implemented in *Caffe* and provides a faster method than the state of art ones yet obtaining a high success rate as the experiments over the *ModelNet10* dataset. This fact enlightens a promising future in real-time 3D recognition tasks.

Following on this work, we plan to improve the inner representation by using adaptable occupancy grids instead of fixed-size ones. In addition, we will integrate the system in an object recognition pipeline for 3D scenes. Our network will receive a point cloud segment of the scene where the object lies, produced by a preprocessing method, and that segments will be used to generate the occupancy grids that will be learned by the system. This implies adapting the system for learning partial views of the objects and dealing with occlusions and scale changes. As an additional feature, we will include pose estimation in that pipeline, all of this with goal of developing an end-to-end 3D object recognition system.

Acknowledgments

This work has been supported by the Spanish Government DPI2013-40534-R[3] grant for the SIRMAVED project [27], supported with Feder funds. This work has also been funded by the grant *”Ayudas para Estudios de Máster e Iniciación a la Investigación”* from the University of Alicante. Experiments were made possible by a generous hardware donation from NVIDIA (Tesla K20).

[3] http://www.iuii.ua.es/SIRMAVED/?idioma=en

References

[1] M. D. Ozturk, M. Ersen, M. Kapotoglu, C. Koc, S. Sariel-Talay, and H. Yalcin, "Scene interpretation for self-aware cognitive robots," in *AAAI-14 Spring Symposium on Qualitative Representations for Robots*, 2014.

[2] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, "3d object recognition in cluttered scenes with local surface features: A survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 11, pp. 2270–2287, 2014.

[3] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3d local feature descriptors," *IJCV-International Journal of Computer Vision*, pp. 1–24, 2015.

[4] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation," *Robotics Automation Magazine, IEEE*, vol. 19, no. 3, pp. 80–91, Sept 2012.

[5] M. Pontil and A. Verri, "Support vector machines for 3d object recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 6, pp. 637–646, 1998.

[6] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class hough forests for 3d object detection and pose estimation," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 462–477.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[10] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3d object classification," in *Advances in Neural Information Processing Systems*, 2012, pp. 665–673.

[11] L. A. Alexandre, "3d object recognition using convolutional neural networks with transfer learning between input channels," in *Proc. the 13th International Conference on Intelligent Autonomous Systems*, 2014.

[12] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[13] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition." IROS, 2015.

[14] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[15] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," *arXiv preprint arXiv:1511.02300*, 2015.

[16] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *International journal of computer vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[17] J. Xiao, J. Hays, K. Ehinger, A. Oliva, A. Torralba *et al.*, "Sun database: Large-scale scene recognition from abbey to zoo," in *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*. IEEE, 2010, pp. 3485–3492.

[18] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[20] J. Bergstra, F. Bastien, O. Breuleux, P. Lamblin, R. Pascanu, O. Delalleau, G. Desjardins, D. Warde-Farley, I. Goodfellow, A. Bergeron *et al.*, "Theano: Deep learning on gpus with python," in *NIPS 2011, BigLearning Workshop, Granada, Spain*, 2011.

[21] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, no. EPFL-CONF-192376, 2011.

[22] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous systems, 2015," *Software available from tensorflow. org*, 2015.

[23] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[24] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[25] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous robots*, vol. 15, no. 2, pp. 111–127, 2003.

[26] A. Dalyac, M. Shanahan, and J. Kelly, "Tackling class imbalance with deep convolutional neural networks," *Imperial College*, pp. 30–35, 2014.

[27] M. Cazorla, J. Garcia-Rodriguez, J. M. C. Plaza, I. G. Varea, V. Matellan, F. M. Rico, J. Martinez-Gomez, F. J. R. Lera, C. S. Mejias, and M. E. M. Sahuquillo, "Sirmaved: Development of a comprehensive monitoring and interactive robotic system for people with acquired brain damage and dependent people," 2015.