IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 13, 2022, accepted January 26, 2022, date of publication February 1, 2022, date of current version February 10, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3148387

3DSliceLeNet: Recognizing 3D Objects Using a Slice-Representation

FRANCISCO GOMEZ-DONOSO[1], FELIX ESCALONA[1], SERGIO ORTS-ESCOLANO[1], ALBERTO GARCIA-GARCIA[2], JOSE GARCIA-RODRIGUEZ[2], (Senior Member, IEEE), AND MIGUEL CAZORLA[1], (Senior Member, IEEE)

[1]Robotics and 3D Vision Laboratory, University Institute for Computer Research, University of Alicante, 03690 Alicante, Spain
[2]Applied Intelligent Architectures, University Institute for Computer Research, University of Alicante, 03690 Alicante, Spain

Corresponding author: Felix Escalona (felix.escalona@ua.es)

This work was supported in part by the Ministerio de Ciencia e Innovación (MCIN)/Agencia Estatal de Investigación (AEI)/10.13039/501100011033 under Grant PID2019-104818RB-I00, and in part by the "European Regional Development Fund (ERDF) A way of making Europe."

ABSTRACT Convolutional Neural Networks (CNNs) have become the default paradigm for addressing classification problems, especially, but not only, in image recognition. This is mainly due to their high success rate. Although a number of approaches currently apply deep learning to the 3D shape recognition problem, they are either too slow for online use or too error-prone. To fill this gap, we propose 3DSliceLeNet, a deep learning architecture for point cloud classification. Our proposal converts the input point clouds into a two-dimensional representation by performing a slicing process and projecting the points to the principal planes, thus generating images that are used by the convolutional architecture. 3DSliceLeNet successfully achieves both high accuracy and low computational cost. A dense set of experiments has been conducted to validate our system under the ModelNet challenge, a large-scale 3D Computer Aided Design (CAD) model dataset. Our proposal achieves a success rate of 94.37% and an Area under Curve (AUC) of 0.978 on the ModelNet-10 classification task.

INDEX TERMS Deep learning, 3D object recognition, convolutional neural networks, Caffe.

I. INTRODUCTION

Object recognition is one of the key problems to be solved for the development of a complete scene understanding system and is the main focus of this work. Although this problem has traditionally been addressed using RGB cameras, in recent years many new approaches have encouraged the use of 3D data. The advent of commodity 3D sensors such as the Microsoft Kinect, and the creation of large, real and synthetic 3D data repositories [1]–[4] have opened new trends for this research problem. In particular, many papers have addressed the problem of 3D shape classification using deep learning techniques and a large number of papers have used Convolutional Neural Networks in the field of 3D object recognition [5]–[7].

While the best results have thus far been obtained with methods based on 2D deep learning, their extension to 3D still presents many problems. For example, the methods that performed best in the ModelNet challenge are mostly based on 2D views. These 2D views are usually obtained as a projection of the 3D data. For example, [8] features a CNN architecture that combines information from multiple views of a 3D shape into a single, compact shape descriptor. This method obtained 90% classification accuracy on the ModelNet40 dataset. However, focusing on 2D visual features could lead to ambiguities in a real scenario. The external part of an object may not capture the internal structure of that object. In addition, most of the best performing techniques in this challenge also rely on the use of multiple 2D views. The main reason why volumetric or 3D rendering approaches do not currently produce as good results as 2D multiple view methods is related to the 3D data discretization process and how the volumetric domain needs to handle large amounts of sparse data. In addition, the cost of handling 3D data is higher than that of processing 2D data, so it is limited by the amount of detail that can be captured. Similar conclusions are presented in [9]. Additionally, as stated in [10], inserting volumetric representations in a deep Convolutional Neural Network (CNN) pipeline requires large amounts of memory and is a very time consuming task.

In this work, which is an extension of the doctoral thesis by Dr. Francisco Gomez-Donoso [11] and our previous work

The associate editor coordinating the review of this manuscript and approving it for publication was Sudipta Roy.

LonchaNet [12], we present an approach that uses multiple 2D views acquired from 3D models applied to 3D object recognition. The proposed 2D representations are based on cross sections of 3D models. The proposed method outperforms our previous method, LonchaNet, which uses different convolutional backbones and more datasets for validation, and also outperforms most of the existing approaches that participated in the ModelNet challenge. We have now obtained 94.37% classification accuracy. Our proposal focuses on learning a discriminative representation that is able to distinguish between most of the categories in the ModelNet dataset. We also contribute a new architecture that utilizes the existing GoogLeNet network [13]. We use three independent GoogLeNet networks for learning features specific to each cross section or slice of the 3D model. Finally, we evaluate the trained model using ModelNet on a subset of the IKEA dataset. We obtained a classification accuracy of 70.45% demonstrating that the trained model was not overfitted to the ModelNet dataset.

The rest of the paper is organized as follows: Section II reviews existing works that use deep learning-based techniques for 3D object recognition. Next, Section III presents the proposed deep learning architecture for 3D object recognition based on 2D renderings from 3D model cross-sections. Section IV shows the experiments and discusses the results obtained using our novel approach. Finally, in Section V, we present our conclusions and directions for future work.

II. RELATED WORKS

Deep learning, in general, and CNNs, in particular, have today surpassed traditional computer vision methods in many tasks, including 3D object recognition [14]–[16]. This exponential and continuous growth has been made possible by three factors: (1) the creation of large-scale 3D object databases, (2) accessible deep learning frameworks for designing, developing, and training CNNs, and (3) the democratization of Graphics Processing Units (GPUs) to accelerate such networks for both inference and training. Although all three were equally vital to the development of deep learning approaches, it is important to note that challenges or benchmarks -associated with data sets- allow researchers to objectively evaluate their proposals against other work. As such, high-quality datasets are gathered around a plethora of competing cutting-edge works striving for the best recognition results. In this regard, the ModelNet database is arguably the most significant challenge, as are the methods that make use of it. ModelNet[1] is a large-scale database of 3D Computer Aided Design (CAD) objects with two subsets or challenges: ModelNet10 and ModelNet40 with 10 and 40 classes respectively. More details on the composition of the dataset and specific information on the challenges are provided in section IV.

This work will focus on ModelNet, and thus, in the following lines we report all the state-of-the-art methods for that challenge, and recent methods not therein included, but whose novelty or results are outstanding. We introduce these methods by grouping them into three broad categories according to their 3D data representations.

Voxelization: Input data is a discretization of the original point cloud, grouping points into different clusters according to a neighborhood criteria. These points serve as an approximation of the original shape. Commonly, every voxel is represented as a binary value, 0 or 1, which indicates the presence of points in the space represented by the voxel.

The seminal work by Wu *et al.* [5] introduced the ModelNet dataset and a Convolutional Deep Belief Network (CDBN) to represent and learn 3D shapes as probability distributions of binary variables on volumetric voxel grids. They achieved 83.50% accuracy, a somewhat inconspicuous percentage for today's standards, but their work paved the way for future research.

Another approach to this problem is presented by Xu and Todorovic in their work ''Beam search for Learning a Deep Convolutional Neural Network of 3D Shapes'' [17], in which a beam search for optimal CNN hyperparameters and architecture is proposed. This system models different network configurations as states, which are connected in a directed graph fashion. The system traverses the graph using a heuristic function that produces the next best state - an improved version of the architecture and the hyperparameters set. This system achieves an accuracy of 88% for the ModelNet-10 classification task and 81.3 for the ModelNet-40 classification task.

On another note, *PointGrid* [18] creates grid cells with a constant number of points with a point quantization technique, saving the point coordinates to improve the representation of the local geometry of the object.

Another important work is the 3D Generative Adversarial Network (GAN) proposed by Wu *et al.* [19], which combines a 3D CNN with a GAN to capture 3D shape descriptors, initially intended to generate or sample 3D objects, which can be effectively reused for classification. By making use of these unsupervised learned descriptors they demonstrated that their model could achieve 91.00% accuracy in the challenge.

Maturana and Scherer proved that bringing together a pure 3D CNN and a volumetric occupancy grid representation was helpful to recognize 3D shapes in an efficient manner. Their proposal, VoxNet [6], achieved a 92.00% success rate on the benchmark.

Other works, such as the Octree-based Convolutional Neural Network (O-CNN) [20] and Octree Generating Network (OGN) [21] combine the use of octree representations with the performance of 3D convolutions to lower memory consumption and improve performance.

The next significant step forward was taken by Sedaghat *et al.* [22] who introduced object orientation prediction, in addition to the class label itself, to increase classification accuracy. Their ORION network is a 3D CNN that produces class labels and orientations as outputs and uses both to contribute to training. By adding orientation

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

estimation as an auxiliary task during training, they were able to learn orientation invariance and raise the accuracy to 93.80%.

Voxception-ResNet (VRN) ensemble mode, presented by Brock *et al.* [23], achieves a significant accuracy of 97.14%. This architecture is based on ResNet [24], but uses inception blocks that are produced by concatenating bottleneck and standard ResNet blocks. A voxelized volumetric input is fed to an ensemble of these VRNs, whose predictions are summed to generate the output.

Finally, in order to approximate the results to real life scenarios, *Par3dnet* [25] used 3D CNNs to perform object recognition over tridimensional partial views of the objects, and made a deep analysis of the easiest and hardest views to classify an object.

2D Projections: In this category of methods, input data is represented as multiple 2D projections of the tridimensional data. They have traditionally been the most common approach, and usually have a 2D CNN to carry out the processing.

DeepPano [7] achieved 85.45% accuracy by converting 3D shapes to panoramic views, using a cylinder projection around their principle axes, and learning them with a CNN specifically designed for that purpose.

Sinha *et al.* [26] propose a system in which a geometry image is created by mapping the mesh surface to a spherical parametrization map, which is then projected onto an octahedron and cut and assembled to create a square. This approach achieves an accuracy of 88.4% and 83.9% in the ModelNet-10 and ModelNet-40 classification tasks, respectively.

Bai *et al.* proposed *GIFT*, a real-time shape matching method that combines projective images of 3D shapes and a CNN to extract features that are subsequently matched and ranked to provide a candidate list; using this approach, they improved slightly with respect to VoxNet reaching a 92.35% recognition rate.

The method described by Johns *et al.* [27] exploited multi-view image sequences to boost accuracy to 92.80%. They used a CNN to independently classify image pairs from sequences, and then classify them again weighting the contribution of each pair.

The Multi-View Convolutional Neural Network (MVCNN) approach, introduced by Su *et al.* [8], used a CNN to learn to classify objects using a collection of rendered views for each one. However, they reported no results for the ModelNet10 challenge. In a subsequent work [28], the authors improve their results by making modifications in the architecture and using shaded images as input.

In the Multi-Loop-View Convolutional Neural Network (MLVCNN) [29], the authors generated a view-loop-shape 3D shape representation structure, which hierarchicalñy represents 3D shapes. They analyzed the view features using a Long Short-Term Memory (LSTM) with Loop Normalization by exploring the relationship between views in each loop.

Finally, *RotationNet* [30] jointly addressed the problem of the pose and object category estimation, using a CNN over a partial set of multi-view images. This network predicts viewpoint-specific category likelihoods corresponding to all predefined discrete viewpoints for each image input, and then selects the object pose that maximizes the object category likelihood.

Point Cloud: 3D data is represented as a raw unordered point cloud. These methods usually extract features by analyzing the neighborhood of every point within a radius.

The most representative proposal in this case is *PointNet++* [31]. This method generates a feature vector for the whole cloud by applying order-invariant transformations to every point, generating local hierarchical features, which that are sampled and grouped, and uses them to segment and classify the scene.

Various proposals are based on the previous architecture. This is the case of *VoteNet* [32], a novel technique based on Hough voting, that uses *PointNet++* layers as the backbone. This approach selects a set of interesting points, with their corresponding features, as seed points to generate clusters of object instances based on their votes. Finally, these clusters are transformed into 3D bounding boxes with their corresponding categories.

Another work, *SplatNet* [33], extends the concept of 2D SPLAT images into 3D. It uses hash tables as a efficient implementation of neighborhood filtering, providing an easy mapping of 2D points into 3D space, bilateral convolutions are then used to extract a set of features.

In the case of *SO-Net* [34], the authors propose a method to guarantee invariance to point permutations. It builds a Self-Organizing Map (SOM) through modelling the spatial distribution of the point cloud and using the neighborhood of every point to extract hierarchical features. As a final step, this method generates a global feature vector for the whole cloud.

Alternative and Fusion Approaches: These approaches use another type of data representation, alternative data transformations, or mix the results of different alternatives.

FusionNet [35] fuses volumetric representations (binary voxel grids) and pixel representations (projected images). The authors use both representations to feed two volumetric CNNs and a MVCNN, achieving 93.11% accuracy.

The Point-Voxel Convolutional Neural Network (PVCNN) [36] combines the sparse representation of the data with voxelized convolutions that increase the performance of the data access and improve the locality of the method. In this work, a new efficient primitive is introduced, Point-Voxel Convolution (PVConv), which converts points into voxel grids, aggregates neighboring points with voxel-based convolutions and transforms them back to points. In order to obtain features with a higher level of detail, the work includes point-based feature transformations.

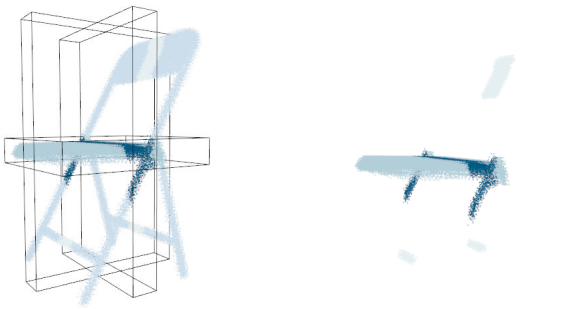
In *NurbsNet* [37], the authors propose a method based on local similarities between surfaces, modeled as nurbs. They fit a nurb surface around the neighborhood of every point,

calculate the similarity score with the pretrained surfaces, select the best similarity score for every part of the object and generate a feature vector to perform the classification.

Hypergraph Neural Networks (HGNN) [38] presents a novel data representation in the form of a hypergraph, using a hyperedge convolution operation to handle the data correlation during representation learning. The authors generate 12 different views of each 3D object in intervals of 30 degrees and create the hypergraph as a probability graph based on the distance between nodes.

Finally, the Voxelized Fractal Descriptor (VFD) [39] proposes a novel global descriptor based on the fractal dimension. This paper proposes the computation of the fractal dimension for every voxel of the object and generates a feature descriptor with the concatenation of the results. The voxel-based computation of the fractal dimension is agnostic to the density of points, number of points in the input cloud, sensor of choice, and noise up to a level.

In light of this literature review, our proposed method presents a novel approach for 3D model recognition that bundles a multi-view object slicing approach, based on Setio *et al.* [40] method for Computed Tomography (CT) images, with a modified version of the GoogLeNet [13] CNN architecture to achieve state-of-the-art performance while keeping the computational cost under control.



(a) Chosen slices of the model (b) Points of every slice

(c) Resultant projections of the three slices

FIGURE 1. Extracting the slices from a point cloud sample.

III. APPROACH

As mentioned above, we propose a method for 3D object recognition using deep learning and 2D CNNs. First, for each sample in the dataset, we take three sections of an object, one for each 3D axis, and project the 3D points onto a plane,

(a) Model slice with scattered pixels (b) Model slice with dilated pixels

FIGURE 2. A comparison of a slice before and after the dilation process. The dilated image provides a more accurate representation the object.

so that we get three images of each sample. Each of these three images that make up a single sample is fed into a Convolutional Neural Network. Our novel deep architecture features three GoogLeNets, one for each image, joined in a layer before the classification layer. The classifier receives the information from the previous three independent networks and performs the classification. This gives us great expressiveness and a high success rate.

A. SLICING A MODEL

3DSliceLeNet takes point clouds as input, but the neural architecture itself uses three images corresponding to three slices. Thus, we first have to extract the slices from the 3D point cloud.

To do this, we load the point clouds and calculate the center point of each axis. We then make a slice in the XY, XZ and YZ planes with a thickness of 5% of the model size. This thickness is empirically determined, allowing the system to capture sufficient data to produce a faithful representation. Points that fall within these sections are isolated and projected onto their planes to generate a 500-pixel image. These images are binary maps in which the background is black and the projected points are white. This process is shown in Figure 1.

Due to the inconsistent point density produced by the sampling process described in section IV-A, the points in some slices are very scattered, so the projection does not faithfully represent the object. To deal with this problem, a post-processing is performed for each projection in which we apply 10 pixel dilations using a square as a structuring element. This post-processing step fills the gap between the sparse points and generates a more adequate representation of the object, as shown in Figure 2 (right).

This process is performed for each sample present in the dataset, so that for each point cloud there are three corresponding images, one per slice.

This slice representation of the object allows us to train and test a 3D recognition system in a 2D way. It provides the high success rate and speed of training and testing that usually characterizes deep image recognition neural networks. Furthermore, it preserves and exploits the 3D information implicitly embodied in the slicing method.

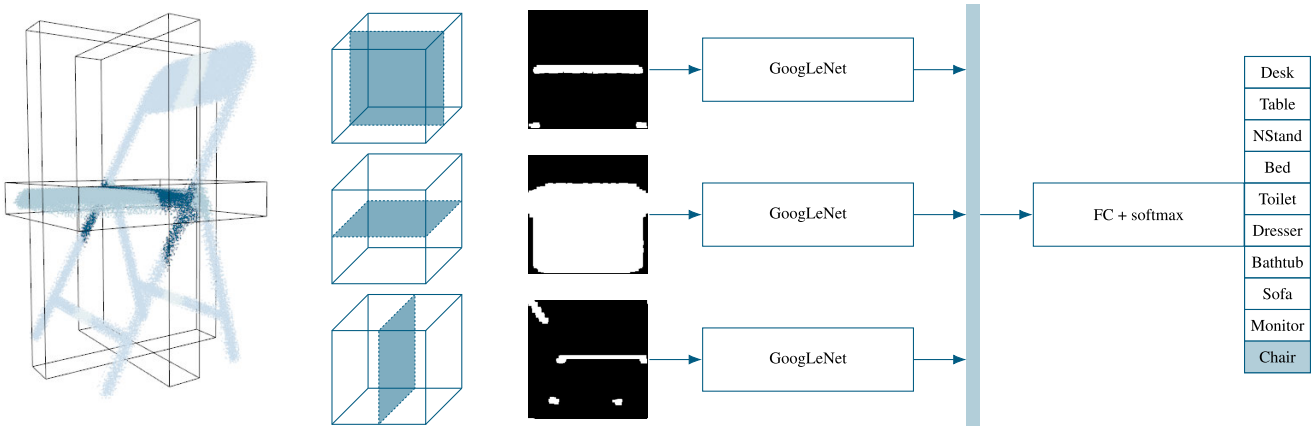


FIGURE 3. 3DSliceLeNet architecture.

B. 3DSliceLeNet ARCHITECTURE

The main architecture of 3DSliceLeNet is composed of three isolated GoogLeNets, which are joined together at the end in a Concatenation Layer prior to a Fully Connected Layer which is the final classifier, as shown in Figure 3.

As mentioned, GoogLeNet is the most advanced deep network for image recognition tasks, providing the highest accuracy in several challenges, which is why we chose it over the other network architectures.

In this architecture, all convolutions, including those of the starting modules, use Rectified Linear Unit (ReLU) activation. The receptive field size in this network is 224×224, taking the RGB channels with mean subtraction, although in the 3DSliceLeNet ensemble we use binary maps without mean normalisation. The GoogLeNet network consists of 22 layers if we consider only the layers with parameter layers (or 27 layers if we also consider clustering layers). The total number of layers (independent building blocks) used for the construction of the network is approximately 100. However, this number depends on the machine learning system used. The use of the average pooling step prior to the classifier is based on [41], although this implementation differs in the use of an extra linear layer. This allows the network to be adapted and tuned for other datasets.

3DSliceLeNet has three independent GoogLeNet (we will refer to them as ''branches''), which learn the features that define an object for each slice. Thus, we force each branch to specialize the filters on particular features of each slice. By isolated, we mean that the hyperparameters of each are different, and are affected independently by the backpropagation stage. Finally, the responses from each branch are concatenated into a single output that is fed to a fully connected layer acting as a classifier.

IV. EXPERIMENTS

In order to validate the precision of 3DSliceLeNet, we tested our approach in the ModelNet challenge. The Princeton ModelNet project intends to provide researchers with a comprehensive clean collection of 3D CAD models for objects and sets a framework to test and compare the different approaches to the 3D object recognition task. First, we describe how to convert this dataset from meshes to a point cloud format, and then we present the methodology and our testing bench. Finally, we show how 3DSliceLeNet performs the ModelNet-10 and ModelNet-40 classification tasks and draw some conclusions. We also use a model obtained by training with the ModelNet-10 dataset to perform inference over the IKEA dataset and describe the results.

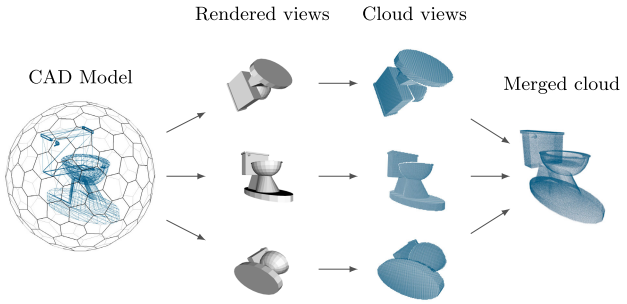


FIGURE 4. From CAD models to point clouds. The object is placed in the center of a tessellated sphere, views are rendered placing a virtual camera in each vertex of the icosahedron, the z-buffer data of those views is used to generate point clouds, and the point clouds are transformed and merged at last.

A. DATASETS FROM MESHES TO POINT CLOUDS

The ModelNet and the IKEA databases [1] provide CAD models in either Object File Format (OFF) or Object File (OBJ) format as polygonal meshes. However, our architecture takes point clouds as an input to generate the slice-based representation – the point cloud representation format is closer to the typical input data provided by consumer depth sensors. To bridge this gap, an adapter or converter stage is performed to shift the OFF and OBJ mesh representation to Point Cloud Data (PCD) clouds. This process involves placing each mesh inside a 3D truncated icosahedron

TABLE 1. ModelNet-10 samples per class distribution.

Class Number	Category	Training Set	Test set
1	Desk	200	86
2	Table	392	100
3	Nighstand	200	86
4	Bed	515	100
5	Toilet	344	100
6	Dresser	200	86
7	Bathtub	106	50
8	Sofa	680	100
9	Monitor	465	100
10	Chair	889	100

(tessellated sphere). A virtual camera is then placed on each vertex pointing to the sphere's center. Raytracing is used to capture a snapshot from each virtual camera, keeping the z-buffer data (depth information) to generate partial point clouds from each view. These point clouds are merged to generate a full model. Additionally, a voxel grid filter is applied to downsample the clouds (uniform point density). Subsequently, those model clouds are sliced and provided as input to the network for training (randomized order), and testing using the corresponding splits provided by ModelNet. Figure 4 illustrates the aforementioned conversion process.

B. METHODOLOGY AND MATERIALS

All timings and results were obtained by conducting the experiments in the following test setup: Intel Core i7-5820K with 32 GiB of Kingston HyperX 2666 MHz and CL13 DDR4 RAM on an Asus X99-A motherboard (Intel X99 chipset). Secondary storage was provided by a Samsung 850 EVO SSD. Additionally, the system included two NVIDIA Tesla K40c GPUs used for training and inference.

The framework of choice was Caffe RC2 running on Ubuntu 14.04.02. It was compiled using CMake 2.8.7, g++ 4.8.2, CUDA 7.5, and cuDNN v3.

We trained and tested 3DSliceLeNet with the ModelNet-10 and ModelNet-40 datasets, as described earlier in the subsection IV-A. It is worth noting that the training and test splits are defined by the dataset itself and that the number of samples per class is not balanced, as seen in Table 1 (both subsets, ModelNet-10 and ModelNet-40 are unbalanced). This fact harms the accuracy of the system, biasing the learning and the classification toward the classes with more number of samples, as stated by [42].

C. RESULTS FOR ModelNet-10

Regarding the parameters that affect the learning process, we trained the architecture with a base learning rate of 0.00001, multiplying the current learning rate by 0.75 every 10000 iterations. To compute the weight update, we use the ADAM [43] solver with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. In the ModelNet-10 experiment, the training process was executed for 20000 iterations, but the best weight set was produced on iteration #18300, which yielded a test accuracy of 94.3709%, the second best score in the leaderboard of the ModelNet10 challenge.

It is also worth noting the low run-time of our architecture. One training iteration with a batch size of 30 samples takes an average of 2.25 seconds. Moreover, classification of new samples only takes 0.0896 seconds.

We cannot compare 3DSliceLeNet with the VRN Ensemble method, which provides a success rate of 97.14% in the ModelNet-10 challenge, because no time measurements were provided in the paper. Nonetheless, we contacted Andrew Brock (author of the mentioned method), who said he did not recall the test time/batch, and all his logs were buried in an external hard drive somewhere and in inference mode would probably take several seconds per batch. In light of this information, VRN Ensemble is, in fact, impractical in a real-time application.

Figure 6 shows the classification rate per class. It can be seen that the classes with the lower success rates are the very same classes that have a lower number of samples, that is, the desk and nightstand classes. Accordingly, we can expect that if we were able to have a balanced dataset, the error rate of these classes would decrease significantly.

Figure 5 shows the confusion matrix of the classification accuracy for the test split of ModelNet-10. The aforementioned confusion matrix, alongside the precision-recall curves presented in Figure 7, confirms the stability and reliability of the system, which fails only on samples that look very similar from a visual perspective. Our system achieves an AUC of 0.978 on this test.

This is the case for the desk and table classes, which the system confuses, but not the other way around. This is arguably caused by the fact that a desk is a type of table, meaning the desks have minor differences (visual features) that go unnoticed in some samples, making such samples hard to distinguish from the table ones, as shown in Figures 8a and 8b. In addition, Table 1 shows that the desk class has a reduced number of samples compared with to classes such as sofa or chair.

In order to corroborate the desk and table ambiguity, a user study was conducted involving humans, who where required to classify random desk and tables samples. This experiment is detailed later in Section IV-D.

In addition, the proposed system fails to distinguish between the nightstand and dresser classes. This kind of problem is common in Convolutional Neural Network architectures because their learned features are mainly based on visual features of an object and, as seen in Figures 8c and 8d, these two classes are visually similar.

Figure 8 shows the ambiguity of the visual features between the desk and table, nightstand and dresser classes and shows the difficulty of the problem.

D. EXPLORING DESK AND TABLE AMBIGUITY WITH HUMANS

Reviewing the desk and table samples of the ModelNet dataset, we noticed there was no substantial contrast in the

Class	Desk	Table	Nstand	Bed	Toilet	Dresser	Bathtub	Sofa	Monitor	Chair
Desk	79	10	2	1	0	2	0	3	1	0
Table	1	96	2	0	0	1	0	0	0	0
Nstand	0	5	86	0	0	9	0	2	1	0
Bed	0	2	0	97	0	0	1	0	0	1
Toilet	0	0	0	1	98	0	0	0	0	1
Dresser	0	0	7	0	0	92	0	10	0	0
Bathtub	0	2	0	4	0	0	94	0	0	0
Sofa	0	0	1	0	0	0	0	99	0	0
Monitor	0	0	0	0	0	0	0	0	100	0
Chair	0	1	0	0	0	0	0	0	0	99

FIGURE 5. Confusion matrix of the classification results achieved by *3DSliceLeNet* after 18300 training iterations using the *ModelNet-10* dataset (solver type is ADAM, learning rate is 0.00001, β_1 is 0.9 and β_2 is 0.999). It is worth noting the confusion between the classes Desk and Table and Nightstand and Dresser. The values shown in the table are percentages.

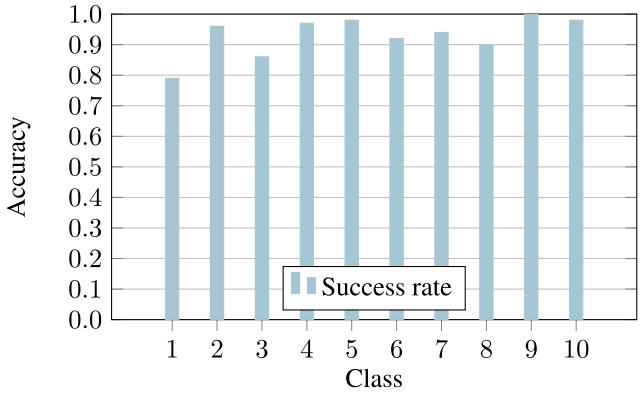


FIGURE 6. Success rate per class for the test split of the ModelNet-10 dataset achieved by 3DSliceLeNet.

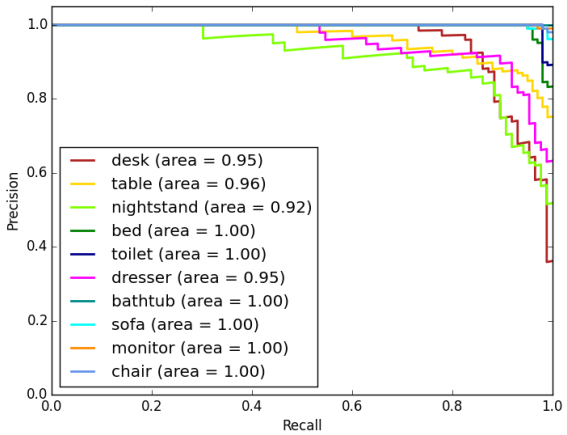


FIGURE 7. Precision-recall curves (classification accuracy) for every class of the ModelNet-10 test split.

visual features that describe each class, so 3DSliceLeNet (and any other visual features-based system) would understandably tend to fail when classifying samples of these classes.

In order to evaluate the scope of the aforementioned desk and table ambiguity, we carried out a new experiment involving humans. This experiment was conducted by displaying 20 random samples of tables and desks (10 of each class) to a set of 9 humans of different professional and academic profiles. Each test subject was asked to classify the samples into either the desk or table class. These results are shown in Table 2. With an overall accuracy of 83.75%, the subjects were unable to successfully guess all samples. Specifically, the humans achieved an accuracy of 88.89% for the desk class and, as expected, 3DSliceLeNet performed similarly with an accuracy of 79.56%.

These results show there are no definitive visual features that allow 3DSliceLeNet to precisely discriminate samples from the desk and table classes. It is therefore prone to fail, as are humans.

When the test participants were questioned about the reasons leding them to classify an object either in the desk or table class, several of them stated that the desks usually have some kind of drawer and tables do not. However, there are samples of tables with drawers and samples of desks without them in the ModelNet dataset, as seen in Figure 8a, and hence this is not a discriminative feature.

The intention of this experiment was to demonstrate the main reason for the lack of accuracy of 3DSliceLeNet on the desk and table classes: their visual features are not sufficiently clear for these two classes, not even by humans.

E. RESULTS FOR ModelNet-40

We also trained and tested our system with the extended version of the ModelNet dataset (40 classes). The neural architecture remained the same with a minor modification: the number of neurons in the output layer was modified from 10 to 40 in order to match the number of classes of the ModelNet-40 dataset. No further modifications were applied to the 3DSliceLeNet architecture.

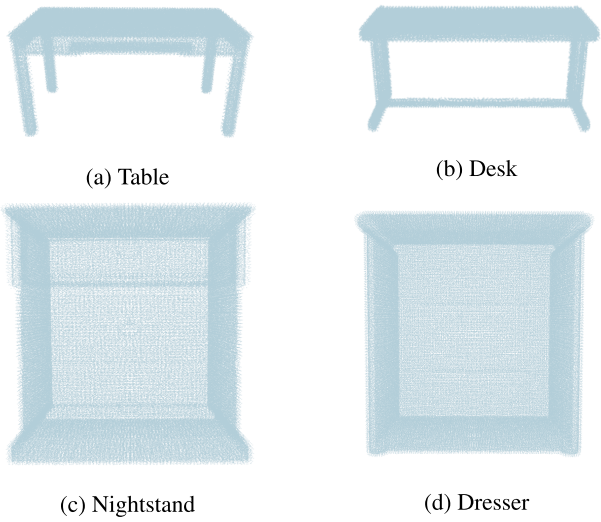


FIGURE 8. Similarity between two objects of different classes: Table and Desk, and Nightstand and Dresser. The point cloud shown in (a) represents an object of the Table class, whilst the point cloud in (b) represents an object whose class is Desk but is misclassified as a Table due to the resemblance. Point clouds (c) and (d) illustrate the same problem with the nightstand and dresser classes.

TABLE 2. Human desk and table classification accuracy. The overall human accuracy of this experiment was 83.75%.

Subject	Desk Acc. (%)	Table Acc. (%)	Total Acc. (%)
1	90	80	85
2	70	80	75
3	100	60	80
4	100	100	100
5	100	90	95
6	70	70	70
7	90	60	75
8	100	80	90
9	80	80	80

The training hyperparameters are the same of those used for the ModelNet-10 experiment: a base learning rate of 0.00001, multiplying the current learning rate by 0.75 every 10000 iterations. To compute the weights update we use the ADAM [43] solver with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

In the ModelNet-40 experiment, the training process was executed for over 250000 iterations, obtaining the best weights set on the iteration #144500 which yielded a test success rate of 79.8529%. The timings for this experiment are approximately the same as those used for the ModelNet-10 experiment: a training iteration of 30 samples took longr than 2.25 seconds, whilst classifying a new sample only took 0.0893 seconds.

Our method is reasonably competitive achieving a high success rate, but it is highly dependent on the orientation of the samples. Whilst in the ModelNet-10 version, the models share the same pose, in the ModelNet-40, they do not. This is caused by the slicing method described in Section III-A, which implicitly captures the clearly counterproductive pose in this case.

Reviewing the confusion matrix obtained in this experiment (Figure 9) and the accuracy per class (Figure 10) we can confirm the overall accuracy of the 3DSliceLeNet architecture. We also observed that several samples across all classes are often wrongly classified as bottles. The misclassified samples of the bowl, cup, flowerpot or stool classes are reasonably comprehensible, as these classes look very similar to a bottle, that is, they have a narrow neck and a wider bottom. As shown in Figure 10, these classes have the lowest success rate. Other classes also share some features, but in a different scale that makes them look very similar once converted to the slice-based representation.

Moreover, we can identify some punctual and scattered confusion errors, such as previously described ambiguity between the desk and table, bookshelf and wardrobe or flowerpot and bottle. These pairs of classes look very similar to each other, leading to a high probability of misclassifying them as explained in Section IV-C.

F. RESULTS FOR THE IKEA DATASET

The IKEA dataset consists of furniture CAD models gathered from the Google 3D Warehouse, alongside aligned RGB images took from Flickr. This dataset contains 7 classes: bed, bookcase, chair, desk, sofa, table and wardrobe. It was originally intended to validate fine 3D pose estimation, but we used the 3D models of the dataset to validate our 3D model recognition system: 3DSliceLeNet.

As we did with the ModelNet dataset, we firstly had to convert the meshes to point clouds following the method explained in Section IV-A and then the obtained point clouds to the sliced representation proposed in Section III-A. We manually aligned the point clouds to match the ModelNet-10 poses. Additionally, several meshes contained more than one object, and so we split them in order to obtain single object meshes. The samples from the bookcase class were removed as this class is not present in the ModelNet-10 dataset. A number of other meshes were removed due to incompatibilities between the given format and our mesh to point cloud method. Finally, the refined IKEA dataset contained 87 samples distributed in 6 classes. The exact number of samples per class can be seen in Table 3.

Finally, we took 3DSliceLeNet and the best ModelNet-10 model, and tested it with the converted IKEA samples achieving an accuracy of 70.4545%. This success rate is promisingly high bearing in mind that we trained our system with the ModelNet dataset and then tested it with a completely different one, which means that the trained model is not overfitted to the ModelNet dataset. Figure 11 shows the results of the classification process.

In this case, the overall accuracy is slightly lower than that achieved by testing over the ModelNet-10 test split shown in Section IV-C. Although 3DSliceLeNet performed well for the table, dresser, sofa or chair classes, it often failed to correctly classify the desk and bed samples. Once more, we found 3DSliceLeNet classifies the desk samples as tables, as in the former experiments. In addition, the desk samples of this

Class	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
airplane	99	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bathtub	0	86	8	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0
bed	0	0	96	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0
bench	0	0	0	60	0	10	0	0	5	0	0	0	5	0	0	0	0	0	0	0	5	0	0	0	0	0	5	0	0	0	0	0	0	10	0	0	0	0	0	0
bookshelf	0	0	0	0	94	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0
bottle	0	0	0	0	1	95	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1
bowl	0	0	0	0	0	30	55	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
car	0	0	0	0	0	34	0	58	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	2	0	0	2	0	0
chair	0	0	0	0	0	0	0	0	97	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	1	0	0	0	0
cone	0	0	0	0	0	10	0	0	0	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	5	0	0
cup	0	5	0	0	0	26	0	0	0	0	26	0	0	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	5	5
curtain	0	0	0	0	5	25	0	0	0	0	0	55	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	5	0	0	0	0	0	0
desk	0	0	1	1	1	0	0	0	0	0	0	0	74	0	0	0	0	0	1	0	0	0	0	1	1	5	0	0	0	1	2	0	0	7	0	0	3	0	0	0
door	0	0	0	0	5	25	0	0	0	0	0	20	0	45	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
dresser	0	0	1	0	1	0	0	0	0	0	0	0	0	0	78	0	1	0	0	0	0	0	0	13	0	0	0	2	0	0	1	0	0	0	0	0	0	0	1	1
flowerpot	0	0	0	0	0	10	0	0	0	0	5	0	0	0	0	10	0	0	0	0	0	0	5	0	0	0	35	0	0	0	0	0	0	0	5	0	30	0	0	
glassbox	0	0	0	0	0	2	0	0	0	0	0	0	0	0	2	0	92	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	1	0	0	0	1	
guitar	0	0	0	3	0	3	0	0	0	0	0	1	0	0	0	0	0	84	0	0	0	0	1	0	0	0	0	5	0	1	0	0	0	0	0	0	1	0	0	
keyboard	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	10	25	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	
lamp	0	0	0	0	0	16	0	0	0	0	0	0	0	0	5	0	0	0	0	68	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	
laptop	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
mantel	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	92	0	0	0	2	0	0	0	0	0	0	0	1	0	0	1	0	0	
monitor	0	0	0	0	0	2	0	0	0	0	0	0	0	0	1	0	0	1	0	1	2	0	89	0	0	1	1	0	0	0	0	0	1	0	0	0	1	0	0	
nstand	0	0	0	0	1	0	0	0	0	0	0	0	0	0	9	0	2	0	0	0	0	0	0	76	0	0	0	0	0	0	0	0	5	0	0	2	0	1	3	
person	0	0	0	0	0	11	0	0	0	0	0	6	0	6	0	0	0	0	0	0	0	0	0	0	33	0	28	0	0	0	0	6	0	0	0	0	0	11	0	
piano	0	0	0	4	1	3	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	2	2	0	0	79	0	2	0	0	0	1	0	1	2	0	0	0	1	
plant	1	0	0	1	2	3	0	0	1	1	0	1	0	0	0	10	0	3	0	2	0	0	1	0	2	0	66	0	0	0	0	0	0	0	1	0	0	4	0	
radio	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	5	0	0	0	0	50	0	0	0	0	0	0	0	0	5	5	0	20
rangehood	0	0	0	0	0	1	0	0	1	2	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	90	0	0	0	0	1	1	0	0	0	0	
sink	0	0	0	0	5	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	80	5	0	0	0	0	0	0	0	0	
sofa	0	0	4	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0.9	0	0	0	0	0	0	0	0	
stairs	0	0	0	5	0	2	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	5	0	0	0	60	0	0	0	0	0	0	0	
stool	0	0	0	0	0	30	0	0	5	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	35	10	0	0	0	5	0	
table	0	0	0	0	0	1	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	82	0	0	0	0	0	1
tent	0	5	0	0	0	5	0	0	0	5	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	75	0	0	0	0	0
toilet	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	97	0	0	0	0
tvstand	0	0	0	1	2	3	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	1	0	3	0	1	0	2	0	0	1	0	0	4	1	0	78	0	0	0
vase	0	0	0	0	2	21	0	0	0	1	3	1	0	0	0	4	1	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	62	0	1
wardrobe	0	0	0	0	10	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	55	1
xbox	0	0	0	0	0	5	0	0	0	0	0	0	0	11	5	0	5	5	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	5	5	0	53

FIGURE 9. Confusion matrix of the classification results achieved by *3DSliceLeNet* and the *ModelNet-40* dataset after 144500 training iterations (solver type is ADAM, learning rate is 0.00001, β_1 is 0.9 and β_2 is 0.999).The values shown in the table are percentages.

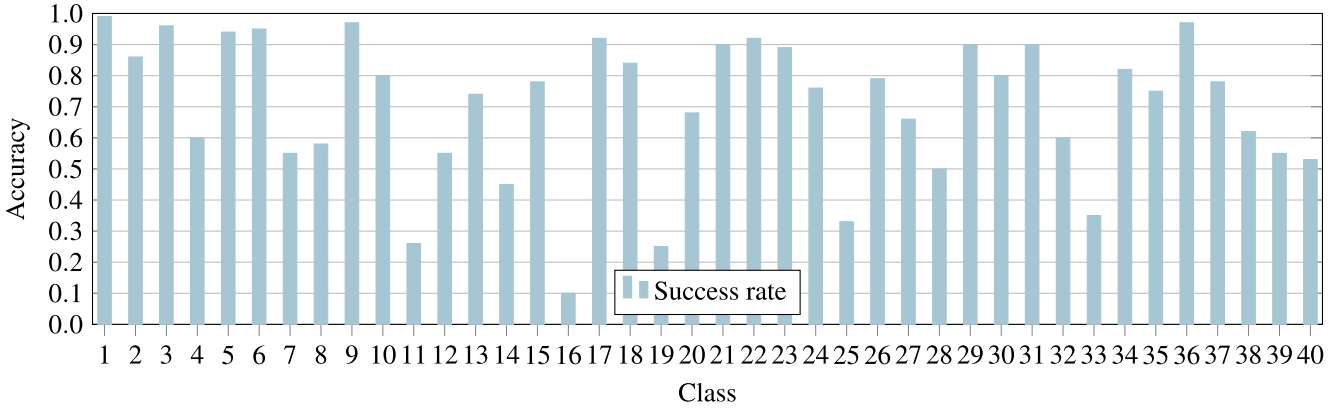


FIGURE 10. Success rate per class for the test split of the ModelNet-40 dataset achieved by 3DSliceLeNet.

dataset are somewhat odd to the system, as they do not only include the desk, but the models also include bookcases or other furniture attached to them, as shown in Figure 12.

The bed class only contains 3 samples. Although it fails to classify all of them, we cannot draw conclusions from this result, due to the reduced size of the set.

It is worth noting that we could have improved classification accuracy by training over this dataset, but this is not the purpose of this experiment. Our target is to state and determine the reliability, stability and generalization capabilities of 3DSliceLeNet, training with a certain dataset and then testing the generated model with a wholly different one.

TABLE 3. Refined IKEA samples per class distribution.

IKEA		
Class Number	Category	# of samples
1	Desk	14
2	Table	39
4	Bed	3
6	Dresser	18
8	Sofa	8
10	Chair	5

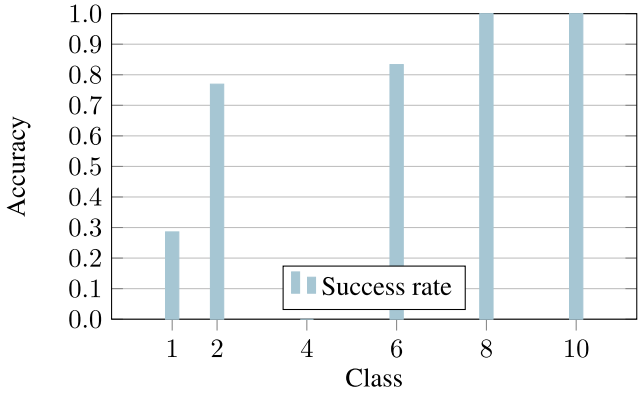


FIGURE 11. Success rate per class for the refined IKEA dataset achieved by 3DSliceLeNet using the best model generated by the training process over the ModelNet-10 dataset. Note that the IKEA dataset does not contain samples for classes 3, 5, 7 or 9.

FIGURE 12. Some samples of desks in the IKEA dataset do not contain only the desk itself but bookcases and other furniture pieces attached to them.

G. RESULTS FOR THE ShapeNet v2 CORE DATASET

ShapeNetCore is a subset of the full ShapeNet dataset, which is an ongoing effort to establish a richly-annotated, large-scale dataset of 3D shapes, with single clean 3D models and manually verified category and alignment annotations. It covers 55 common object categories with about 51,300 unique 3D models. Table 4 shows the number of samples per category.

The samples of this dataset come in a mesh format, and so they had to be converted to point clouds following the method described in Section IV-A in order to be used with 3DSliceLeNet.

Next, 3SliceLeNet was trained on this dataset. This experiment was again conducted on the machine described in Section IV-B. The test split ratio was 20% of the dataset. Once the training process was done, the test yielded an accuracy of 88.45%. The overall accuracy, although high, is not as high as that achieved in the ModelNet-10 test. Both datasets are manually aligned but, in this case, the number of samples per class is highly unbalanced. In fact, some classes contain fewer than 100 samples, whilst others contain more than 8000. This causes the learning process to skew to the categories with more samples, as it is more likely to make a correct prediction if it predicts the class with more samples. This effect harms the overall accuracy. As seen in Figure 13, the classes with lower accuracy are those with a lower number of samples (Table 4). Figure 14 shows the confusion matrix for this experiment. The diagonal confirms that the algorithm performs as expected, with minor failing cases. For instance, almost every *cellphone* sample is classified as *telephone*. This is understandable as the *telephone* class includes samples of *cellphone* or isolated telephone handsets that are very similar to *cellphone*. Furthermore, some examples of *microphone* are classified as *lamp*, which is also understandable as both classes share common visual features, namely a tall post with a larger artifact on top. Finally, the accuracy of the *tower* class is low, as the system tends to classify its samples as *lamp*. Once again, *tower* and *lamp* share common visual features, namely a thin tall structure.

TABLE 4. ShapeNet v2 Core samples per class distribution.

Class	Category	Samples	Class	Category	Samples
1	Airplane	4045	29	Jar	596
2	Trashcan	343	30	Knife	412
3	Handbag	83	31	Lamp	2317
4	Handbasket	113	32	Laptop	460
5	Bathtub	856	33	Speaker	1597
6	Bed	233	34	Mailbox	94
7	Bench	1813	35	Microphone	67
8	Bird House	73	36	Microwave	152
9	Bookshelf	452	37	Motorcycle	337
10	Bottle	498	38	Mug	214
11	Bowl	186	39	Piano	239
12	Bus	939	40	Pillow	96
13	Cabinet	1571	41	Pistol	307
14	Camera	113	42	Pot	599
15	Tincan	108	43	Printer	166
16	Cap	56	44	Remote	66
17	Car	3514	45	Rifle	2371
18	Cellphone	830	46	Rocket	85
19	Chair	6778	47	Skate	151
20	Clock	651	48	Sofa	3173
21	Keyboard	65	49	Stove	218
22	Dishwasher	93	50	Table	8435
23	Display	1091	51	Telephone	1088
24	earphone	73	52	Tower	133
25	Faucet	744	53	Train	389
26	File	298	54	Watercraft	1938
27	Guitar	797	55	Washer	169
28	Helmet	162			

H. TESTING OTHER BRANCH ARCHITECTURES

All the experiments thus far presented were executed on a topology that features three GoogLeNet branches, but other architectures were also considered. This subsection compiles and details these experiments. They evidence that the chosen GoogLeNet architecture outperforms several others and

justifies the inclusion of GoogLeNet in the 3DSliceLeNet topology.

All the experiments were conducted with the parameters detailed in Section IV-B and trained and tested on the ModelNet-10 dataset.

First, ResNet50 [24] was tested. This architecture introduces the ''residual'' term, which consists of the aggregation of the input image to the output image of a convolution block. As a result, the output of a convolution block can be seen as the input image where the features activated by the filters are highlighted. In contrast, the output of a convolution layer in a default convolutional neural network is only the result of the neuron activation. If a neuron is not triggered on a certain region of the input image, the output remains with lower activation values. When the network computes the weight updates in the backpropagation stage, the values on non-activated regions lead to very low updates, eventually even provoking no update at all, which causes the learning to stop. This issue is known as the vanishing gradient problem. The inclusion of the ''residual'' term helps tackle the vanishing gradient problem and allows the creation of even deeper architectures.

A 3DSliceLeNet with ResNet50 branches was trained and tested on the ModelNet-10 dataset. This topology achieved an accuracy of 92.2822% and a runtime of 0.1053 seconds on inference mode. The confusion matrix of this experiment is shown in Table 5.

In addition, the Xception [44] architecture was tested. This architecture introduces the depthwise separable convolution, which consists in applying convolutions across channels and then a 1×1 convolution. This feature makes the network learn spatial dependencies and relations across channels.

A 3DSliceLeNet with Xception branches was trained and tested on the ModelNet-10 dataset. This topology achieved an accuracy of 91.9514% and a runtime of 0.0942 seconds. The confusion matrix of this experiment is shown in Table 6.

Lastly, the VGG16 [45] architecture was also considered for inclusion in the 3DSliceLeNet topology. This architecture features a stack of convolutional layers (with a different depth in different architectures) and is followed by three fully connected layers: the first two have 4096 neurons each, the third performs classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks. To integrate VGG16 in 3DSliceLeNet, the fully connected layers were removed so the outputs of the last convolution blocks of the three branches are fed to the 3DSliceLeNet classification block.

A 3DSliceLeNet with VGG16 branches was trained and tested on the ModelNet-10 dataset. This topology achieved an accuracy of 90.6284% and a runtime of 0.2248 seconds. The confusion matrix of this experiment is shown in Table 7.

Although the accuracy gain is marginal, the 3DSliceLeNet with VGG16 branches is totally discarded because of its elevated number of parameters (over 400.000). This causes the network to run much slower and takes more memory to work than the other tested architectures. What is more, the main novelty of Xception lies in the intra-channel convolutions. Since the point cloud projections are binary maps, it cannot take advantage of this feature, so this architecture was discarded regardless of its accuracy. Finally, 3DSliceLeNet with ResNet50 branches performed similarly to the GoogLeNet branch incarnation, yet its accuracy is slightly lower. As described, all the considered architectures achieved similar accuracies but GoogLeNet was the best performer on both inference time and accuracy, and was thus chosen to be part of the 3DSliceLeNet final topology.

I. IMPACT OF THE NUMBER OF SLICES ON THE ACCURACY AND RUNTIME

As stated, using only three slices to feed our architecture involves discarding much potentially useful information. However, our chosen topology for 3DSliceLeNet with GoogLeNet branches cannot be tested any further due to the memory limitations of our current hardware set up. In order to establish the gain when increasing the number of slices, the branches of the 3DSliceLeNet topology were replaced by the simpler LeNet5 [46], a much shallower and naive architecture, but which is also less memory greedy. In this way, up to six slices fit in our system with no memory problems. These experiments were intended to show the improvement in accuracy by using more than three slices.

The experiments were again conducted with the parameters detailed in Section IV-B and trained and tested on the ModelNet-10 dataset.

First, a 3DSliceLeNet with three LeNet5 branches was tested. This topology achieved a test accuracy of 82.24% and a runtime of 0.0122 seconds at inference time.

A 3DSliceLeNet with six LeNet5 branches was then tested. This topology achieved an accuracy of 82.02% and a runtime of 0.0178 seconds at inference time.

As expected, the overall accuracy dropped compared to the 3DSliceLeNet chosen topology as the LeNet5 architecture is shallower than the GoogLeNet one. The aim of these experiments, however, was not to improve the accuracy but to ascertain how much gain is yielded by using more slices to classify 3D objects.

These experiments also do not allow us to conclude whether the inclusion of more slices could lead to an improvement. In fact, the accuracy of 3DSliceLeNet with three or six slices is approximately the same while the inference time is increased by a 45%. Nonetheless, this conclusion is not definitive. It is worth recalling that the LeNet5 architecture expressiveness is limited compared to deeper architectures, such as GoogLeNet. Nevertheless, there would likely be room to improve the classification accuracy when feeding more slices if the architectures in the branches were more powerful.

J. DISCUSSION

Using our current architecture, we reached a top-5 place in the leaderboard of the challenge, with an accuracy of 94.37% in

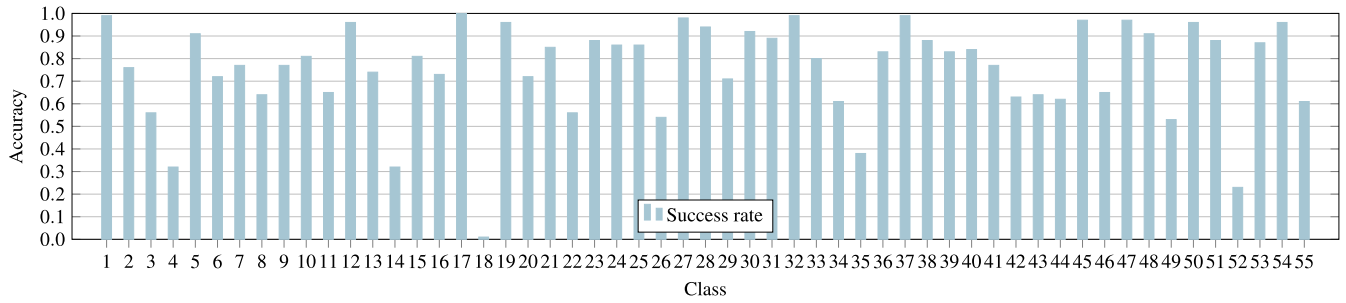


FIGURE 13. Success rate per class for the test split of the ShapeNet v2 Core dataset achieved by 3DSliceLeNet.

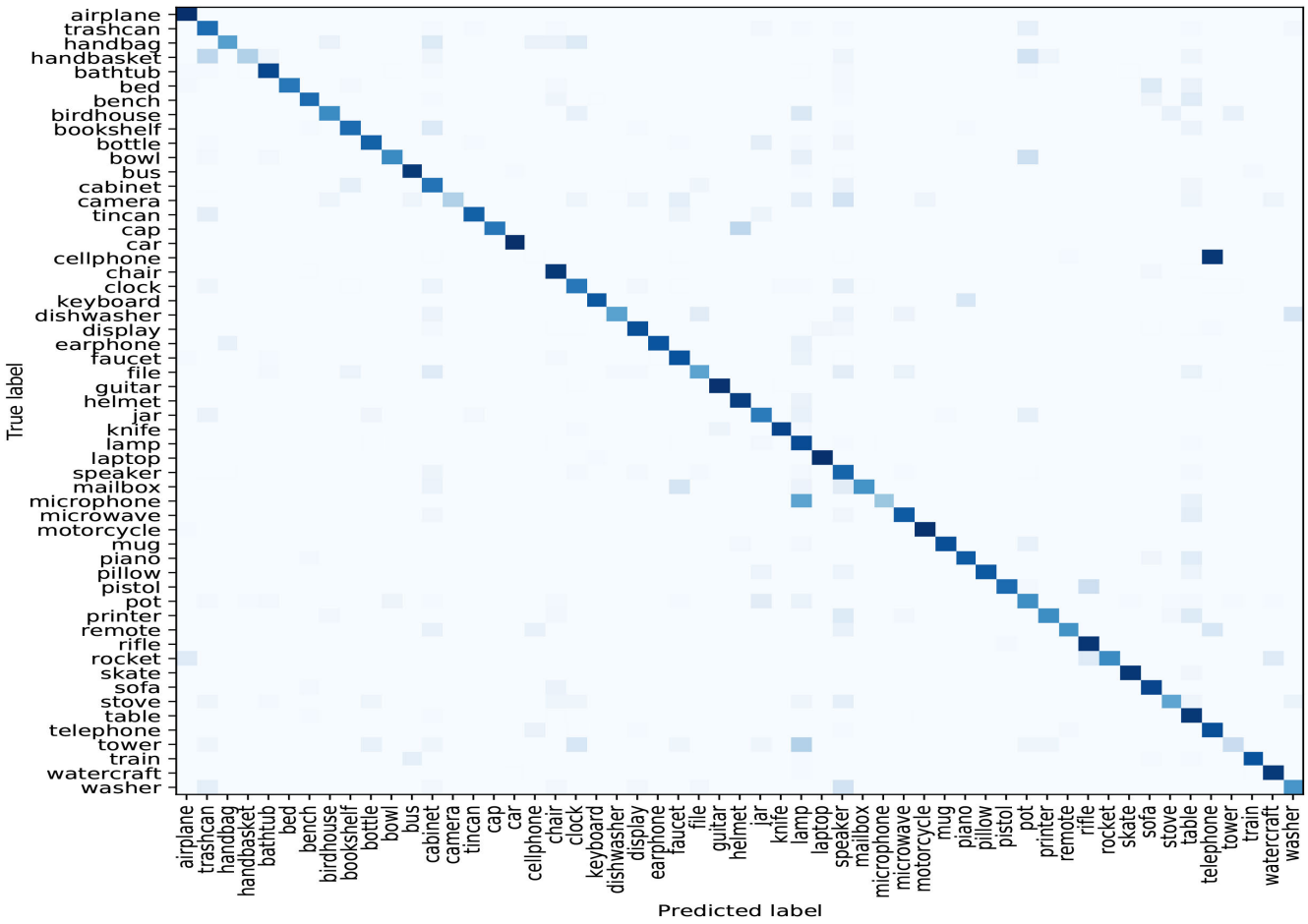


FIGURE 14. Confusion matrix of the classification results achieved by *3DSliceLeNet* and the *Shapenet v2 Core* dataset after 144500 training iterations (solver type is ADAM, learning rate is 0.00001, β_1 is 0.9 and β_2 is 0.999). Darker shades of blue represent higher values.

the ModelNet-10 classification task and reached an accuracy of 79.85% in the ModelNet-40 classification task.

Nevertheless, we also tested another approach for this problem. This approach consisted in concatenating several slices for each 3D axis into a single image, thus generating $3 \times N$ slices for each sample. Each slice is understood as a tile, and so this approach relies on images of a high resolution. If more slices are considered, images of a grater resolution are generated. This approach is much more memory consuming than the other approach tested with 3 convolutional branches. We converted the meshes to this 2D representation and fed it into a state-of-the-art GoogLeNet. We noticed that better classification results were obtained as we increased the N value, i.e., the number of slices per 3D axis. This approach was tested with 3×3, 3×5 and 3×7 slices observing an improvement on each test over the one before. We could not test any further due to memory limitations.

TABLE 5. Confusion matrix of the classification results achieved by *3DSliceLeNet* with ResNet50 branches using the *ModelNet-10* dataset (solver type is ADAM, learning rate is 0.00001, β_1 is 0.9 and $beta_2$ is 0.999). The values shown in the table are percentages.

Class	Desk	Table	Nstand	Bed	Toilet	Dresser	Bathtub	Sofa	Monitor	Chair
Desk	77	8	7	0	2	0	0	2	2	1
Table	1	97	2	0	0	1	0	0	0	0
Nstand	0	3	92	0	0	3	0	0	0	1
Bed	0	2	1	94	3	0	0	0	0	0
Toilet	0	0	0	3	97	0	0	0	0	0
Dresser	0	0	21	0	0	79	0	0	0	0
Bathtub	0	2	0	8	2	0	88	0	0	0
Sofa	0	0	1	0	1	0	0	98	0	0
Monitor	0	0	2	0	0	0	0	1	98	0
Chair	0	0	2	1	0	0	0	0	0	97

TABLE 6. Confusion matrix of the classification results achieved by *3DSliceLeNet* with Xception branches using the *ModelNet-10* dataset (solver type is ADAM, learning rate is 0.00001, β_1 is 0.9 and $beta_2$ is 0.999). The values shown in the table are percentages.

Class	Desk	Table	Nstand	Bed	Toilet	Dresser	Bathtub	Sofa	Monitor	Chair
Desk	74	13	1	3	0	1	0	5	1	1
Table	1	99	0	0	0	0	0	0	0	0
Nstand	0	2	88	0	0	7	0	0	1	1
Bed	0	2	0	98	0	0	0	0	0	0
Toilet	0	0	0	2	97	0	0	0	0	1
Dresser	0	0	16	0	0	83	0	0	1	0
Bathtub	0	0	0	8	4	0	88	0	0	0
Sofa	0	0	0	0	0	0	0	100	0	0
Monitor	0	1	1	0	0	0	1	0	97	0
Chair	0	0	0	0	0	0	0	0	0	100

TABLE 7. Confusion matrix of the classification results achieved by *3DSliceLeNet* with VGG16 branches using the *ModelNet-10* dataset (solver type is ADAM, learning rate is 0.00001, β_1 is 0.9 and $beta_2$ is 0.999). The values shown in the table are percentages.

Class	Desk	Table	Nstand	Bed	Toilet	Dresser	Bathtub	Sofa	Monitor	Chair
Desk	84	2	2	0	0	1	0	7	1	3
Table	19	77	2	0	0	0	0	0	0	2
Nstand	0	1	64	0	0	14	0	2	1	17
Bed	0.1	0	0	92	0	0	1	0	0	6
Toilet	0	0	0	1	93	0	0	0	0	6
Dresser	0	0	13	1	0	79	5	2	0	0
Bathtub	0	0	0	4	2	0	90	4	0	0
Sofa	0	0	0	0	0	0	0	100	0	0
Monitor	0	0	0	0	0	1	0	2	95	2
Chair	0	0	0	0	0	0	0	0	0	100

These findings lead us to think that if we were able to feed 3DSliceLeNet with more slices, we could expect an improvement in the accuracy rate. However, we cannot test this case due to the memory limitations of our hardware (GPU memory consumption above 12 GBytes).

V. CONCLUSION

This paper presents a novel architecture for 3D object recognition, 3DSliceLeNet. Our system takes three slices of the input point cloud (one per 3D axis), projects the points to a plane, generating three images, and uses these images as an

input for the proposed deep network. The architecture consists of three independent GoogLeNet branches, the activations of which are concatenated and fed into a fully connected layer. Each of these branches learns particular features of a slice. This method allows us to take advantage of the fast 2D computation whilst preserving the 3D information. 3DSliceLeNet achieved a success rate of 94.37% in the ModelNet-10 classification task and reached an accuracy of 79.85% in the ModelNet-40 classification task, while providing extremely fast computation times: once the model is trained, classifying a 3D object only takes 0.0896 seconds.

VI. FUTURE WORK

Following on from this work, we plan to address the generalization problem caused by inconsistent poses across the models. This problem was revealed during the ModelNet-40 experiments. This issue can be addressed by applying data augmentation methods, amd so further research in this line should be conducted.

In addition, we plan to extend this system to a 3D object recognizer for point clouds captured in the real world with low cost depth sensors, such as the Microsoft Kinect device. This involves new challenges, as these sensors do not obtain a whole point cloud representation of the scene, but only a partial view of. The real world also presents certain peculiarities that affect the classification process, such as dealing with complete scenes, filled with different objects, and with occlusion problems.

In addition, we plan to test a new version of 3DSliceLeNet that uses several slices per 3D axis. Hence, we are currently exploring methods to circumvent GPU memory limitations.

REFERENCES

[1] J. J. Lim, H. Pirsiavash, and A. Torralba, "Parsing IKEA objects: Fine pose estimation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2992–2999.

[2] S. Song, S. P. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D scene understanding benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 567–576.

[3] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese, "ObjectNet3D: A large scale database for 3D object recognition," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 160–176.

[4] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q.-X. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," *CoRR*, vol. abs/1512.03012, Dec. 2015.

[5] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.

[6] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 922–928.

[7] B. Shi, S. Bai, Z. Zhou, and X. Bai, "DeepPano: Deep panoramic representation for 3-D shape recognition," *IEEE Signal Process. Lett.*, vol. 22, no. 12, pp. 2339–2343, Dec. 2015.

[8] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 945–953.

[9] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Apr. 2016, pp. 5648–5656.

[10] A. Garcia-Garcia, F. Gomez-Donoso, J. Garcia-Rodriguez, S. Orts-Escolano, M. Cazorla, and J. Azorin-Lopez, "PointNet: A 3D convolutional neural network for real-time object class recognition," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 1578–1584.

[11] F. Gomez-Donoso, "Contributions to 3D object recognition and 3D hand pose estimation using deep learning techniques," Ph.D. dissertation, Dept. Comput. Sci. Artif. Intell., Univ. Alicante, Alicante, Spain, 2020.

[12] F. Gomez-Donoso, A. Garcia-Garcia, J. Garcia-Rodriguez, S. Orts-Escolano, and M. Cazorla, "LonchaNet: A sliced-based CNN architecture for real-time 3D object recognition," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 412–418.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[14] W. Wang, Y. You, W. Liu, and C. Lu, "Point cloud classification with deep normalized Reeb graph convolution," *Image Vis. Comput.*, vol. 106, Feb. 2021, Art. no. 104092.

[15] W. Li, F.-D. Wang, and G.-S. Xia, "A geometry-attentional network for ALS point cloud classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 164, pp. 26–40, Jun. 2020.

[16] C. Wen, L. Yang, X. Li, L. Peng, and T. Chi, "Directionally constrained fully convolutional neural network for airborne LiDAR point cloud classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 162, pp. 50–62, Apr. 2020.

[17] X. Xu and S. Todorovic, "Beam search for learning a deep convolutional neural network of 3D shapes," 2016, *arXiv:1612.04774*.

[18] T. Le and Y. Duan, "PointGrid: A deep network for 3D shape understanding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9204–9214.

[19] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," 2016, *arXiv:1610.07584*.

[20] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-CNN: Octree-based convolutional neural networks for 3D shape analysis," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–11, 2017.

[21] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2088–2096.

[22] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox, "Orientation-boosted voxel nets for 3D object recognition," 2016, *arXiv:1604.03351*.

[23] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," 2016, *arXiv:1608.04236*.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*.

[25] F. Gomez-Donoso, F. Escalona, and M. Cazorla, "Par3DNet: Using 3DCNNs for object recognition on tridimensional partial views," *Appl. Sci.*, vol. 10, no. 10, p. 3409, May 2020.

[26] A. Sinha, J. Bai, and K. Ramani, *Deep Learning 3D Shape Surfaces Using Geometry Images*. Cham, Switzerland: Springer, 2016, pp. 223–240, doi: 10.1007/978-3-319-46466-4_14.

[27] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition," 2016, *arXiv:1605.08359*.

[28] J.-C. Su, M. Gadelha, R. Wang, and S. Maji, "A deeper look at 3D shape classifiers," in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, 2018, pp. 1–16.

[29] J. Jiang, D. Bao, Z. Chen, X. Zhao, and Y. Gao, "MLVCNN: Multi-loop-view convolutional neural network for 3D shape retrieval," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 8513–8520.

[30] A. Kanezaki, Y. Matsushita, and Y. Nishida, "RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5010–5019.

[31] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[32] C. R. Qi, O. Litany, K. He, and L. Guibas, "Deep Hough voting for 3D object detection in point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9277–9286.

[33] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz, "SPLATNet: Sparse lattice networks for point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2530–2539.
[34] J. Li, B. M. Chen, and G. H. Lee, "SO-Net: Self-organizing network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9397–9406.
[35] V. Hegde and R. Zadeh, "FusionNet: 3D object classification using multiple data representations," *CoRR*, vol. abs/1607.05695, Nov. 2016.
[36] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-voxel CNN for efficient 3D deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 963–973.
[37] F. Escalona, D. Viejo, R. B. Fisher, and M. Cazorla, "NurbsNet: A nurbs approach for 3D object recognition," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–7.
[38] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 3558–3565.
[39] J. F. Domenech, F. Escalona, F. Gomez-Donoso, and M. Cazorla, "A voxelized fractal descriptor for 3D object recognition," *IEEE Access*, vol. 8, pp. 161958–161968, 2020.
[40] A. A. A. Setio, F. Ciompi, G. Litjens, P. Gerke, C. Jacobs, S. J. van Riel, M. M. W. Wille, M. Naqibullah, C. I. Sánchez, and B. van Ginneken, "Pulmonary nodule detection in CT images: False positive reduction using multi-view convolutional networks," *IEEE Trans. Med. Imag.*, vol. 35, no. 5, pp. 1160–1169, May 2016.
[41] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, Dec. 2013.
[42] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009, doi: 10.1109/TKDE.2008.239.
[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, Dec. 2014.
[44] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *CoRR*, vol. abs/1610.02357, Oct. 2016.
[45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, Sep. 2014.
[46] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

FRANCISCO GOMEZ-DONOSO received the B.S. degree in computer science and the master's degree in robotics and automatic from the University of Alicante, Spain, in 2014 and 2015, respectively, where he is currently pursuing the Ph.D. degree in computer science. Regarding his experience as a Scientist, he has published more than 35 papers in high-impact journals and conferences. His main research interests include human–computer interaction, deep learning, machine learning, and tridimensional data processing.

FELIX ESCALONA was born in Alicante, Spain, in 1994. He received the B.S. degree in computer science and the M.S. degree in robotics from the University of Alicante, Spain, in 2016 and 2017, respectively, where he is currently pursuing the Ph.D. degree in computer science, with an FPU Scholarship. His research interests include 3D object recognition, segmentation and scene understanding, as well as domestic and social robotics. He received the Extraordinary End-of-Degree Prize from the University of Alicante.

SERGIO ORTS-ESCOLANO received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the University of Alicante, Spain, in 2008, 2010, and 2014, respectively. He is currently an Assistant Professor with the Department of Computer Science and Artificial Intelligence, University of Alicante. Previously, he was a Researcher at Microsoft Research, where he was one of the leading members of the Holoportation Project (virtual 3D teleportation in real-time). He has authored more than 50 publications in journals and top conferences, such as CVPR, SIGGRAPH, 3DV, BMVC, *Neurocomputing*, *Neural Networks*, and *Applied Soft Computing*. His research interests include computer vision, 3D sensing, real-time computing, GPU computing, and deep learning. He is also member of European Networks, such as HiPEAC and Eucog.

ALBERTO GARCIA-GARCIA received the Ph.D. degree in machine learning and computer vision from the University of Alicante, in 2019. He was a Postdoctoral Researcher at the Institute of Space Sciences (ICE-CSIC, Barcelona) working on the MAGNESIA ERC Consolidator Project. He was an Intern at NVIDIA Research/Engineering, Facebook Reality Labs, and Oculus Core Tech. He is currently a Researcher at Facebook Reality Labs, Zürich. His main research interests include deep learning, virtual reality, 3D computer vision, and parallel computing on GPUs.

JOSE GARCIA-RODRIGUEZ (Senior Member, IEEE) received the Ph.D. degree with specialization in computer vision and neural networks from the University of Alicante, Spain. He is currently a Full Professor with the Department of Computer Technology, University of Alicante. His research interests include computer vision, machine learning, pattern recognition, robotics, man-machine interfaces, ambient intelligence, and parallel and multi-core architectures.

MIGUEL CAZORLA (Senior Member, IEEE) received the Ph.D. degree in computer engineering from the University of Alicante, in 2000.

In 1995, he started as an Assistant Professor at the University of Alicante. He was a Computer Engineer with the University of Alicante, in 1995. Since 2017, he has been a Full Professor with the University of Alicante. He has completed several stays at foreign institutions (Carnegie Mellon University, University of Sydney, and The University of Edinburgh). He has published more than 50 articles indexed in JCR (with more than 20 in Q1) and more than 100 publications in national and international conferences. He has supervised 11 Ph.D. theses and is a principal investigator in several national projects (CICYT, Challenges), as well as having completed multiple transfer contracts with the industry. His research line has always focused on computer vision. From the beginning, he applied these skills to try to solve robotic tasks. Almost since his inception in research, he worked in the processing of 3D data. In recent years, he has diversified his lines to apply deep learning techniques to different areas (medical image, object recognition, depth estimation, and identification of traffic objects). All his research in recent years has focused on social robotics, that is, applying these techniques to help dependent persons.

Dr. Cazorla is a member of different program committees of national and international conferences.