Learning and Nonlinear Models - Journal of the Brazilian Society on Computational Intelligence (SBIC), Vol. 22, Iss. 1, pp. 45-60, 2024
© Brazilian Computational Intelligence Society

LIDAR3DNETV2: SMART APPROACH TO CLASSIFY 3D OBJECTS BASED ON REAL POINT CLOUDS

Iágson Carlos L. Silva , José Carlos L. Moreira , Francisco Hércules dos S. Silva
Postgraduate Program in Electrical Engineering, Federal University of Ceará, Fortaleza, Ceará, Brazil
Postgraduate Program in Computer Science, Federal Institute of Ceará, Fortaleza, Ceará, Brazil
{iagsoncarlos,jose.carlos,herculessilva}@lapisco.ifce.edu.br

Suane Pires P. da Silva Pedro P. Rebouças Filho , Pedro H. Feijó de Sousa
Postgraduate Program in Teleinformatics Engineering, Federal University of Ceará, Fortaleza, Ceará, Brazil
Postgraduate Program in Computer Science, Federal Institute of Ceará, Fortaleza, Ceará, Brazil
{suanepires,pedrofeijo}@lapisco.ifce.edu.br, pedrosarf@ifce.edu.br

Resumo – As nuvens de pontos geradas em simuladores evitam custos com tempo de coleta e fornecem uma quantidade elevada e organizada de nuvens de pontos, cenário ideal para redes de aprendizagem profunda. No entanto, essas redes têm limitações quando aplicadas a nuvens de pontos reais. Este trabalho propõe um método baseado em perceptron multicamadas para classificar objetos 3D com base em nuvens de pontos reais obtidas usando sensores LiDAR. O método inclui uma etapa de pré-processamento que normaliza e ajusta as nuvens de pontos no plano cartesiano 3D para superar discrepâncias na distribuição de pontos. Além disso, um conjunto de dados foi criado e o conjunto de dados ModelNet é usado para fins de comparação. A rede neural proposta, Lidar3DNetV2, alcançou 98,47% e 125 μs em precisão e tempo de teste com dados reais, respectivamente. A etapa de pré-processamento proporcionou um aumento significativo no desempenho do classificador. Por fim, o método proposto possui um desempenho superior as demais do redes do estado da arte considerando nas nuvens de pontos reais.
Palavras-chave – Nuvem de Pontos, Aprendizado de Máquinas, Classificação de Objetos 3D, LiDAR.

Abstract – Point clouds generated in simulators avoid collection time costs and provide a high and organized amount of point clouds, an ideal scenario for deep learning networks. However, these networks have limitations when applied to real point clouds. This work proposes a multilayer perceptron-based method to classify 3D objects based on real point clouds obtained using LiDAR sensors. The method includes a pre-processing step that normalizes and adjusts the point clouds in the 3D Cartesian plane to overcome discrepancies in the point distribution. Furthermore, we created a dataset and used the ModelNet dataset for comparison purposes. The proposed neural network, Lidar3DNetV2, achieved 98.47% and 125 μs in accuracy and test time with real data, respectively. The pre-processing step provided a significant increase in the classifier's performance. Finally, the proposed method performs better than other state-of-the-art networks considering real point clouds.
Keywords – Point Cloud, Machine Learning, 3D Object Classification, LiDAR.

1. Introduction

Research on different data representations has driven the development of advanced techniques for capturing, manipulating, and visualizing these data. Within the broad spectrum of three-dimensional data representations, point clouds stand out in computer graphics and computer vision, thanks to the ease with which they can be manipulated and visualized [1]. Among the various 3D data representations, point clouds are widely used in computer graphics and computer vision due to their user-friendly manipulation and visualization [1].In addition, 3D space point clouds are applied to improve the user experience, allowing navigation and orientation within 3D models with details that enable the identification of local features of interest [2]. Thus, these point clouds are used in vehicle navigation applications [3–5], including free space detection [6, 7], road segmentation [8–10], and road anomaly detection [11].

With the advancement of computing resources, such as GPUs, and access to 3D data from depth sensors, deep learning on 3D data has become more prevalent. Recent research explores processing to segment and classify 3D objects in real-time [12–14]. However, point clouds are sparse, unstructured, and disordered [15]. In this context, some works propose datasets, in addition to developing methods to manipulate, segment, and classify point clouds, for example, ShapeNet [16], ModelNet40, and nuScenes [5]. In this work, we use real point clouds generated during research to classify 3D objects using machine learning techniques. The authors propose artificial neural network approaches based on modeling or point grouping strategies in the literature. The SO-Net network, for example, models the spatial distribution of the point cloud by creating a Self-Organizing Map [17]. Semantic Segmentation of 3D Point Clouds (SEGCloud) takes point clouds of urban locations and classifies them using the 3D voxel grid and 3D neighborhood features (multiscale). This technique clusters a set of points and generates a voxel analogous to the pixel present in images [18]. Landrieu and Simonovski [19] organize the 3D point cloud using deep geometry learning, which is one of the challenges, especially for 3D objects [20–23]. Due to the rapid development of deep learning, these models can perform classification and shape recognition tasks for different types of 3D geometries [20, 21, 23, 24].

Table 1: Features of 3D Point Cloud Classification Works.

Dataset and Neural Networks	Features
ShapeNet	Dataset for 3D objects, supports various categories and has a large collection
ModelNet40	Dataset for 3D object classification, containing various synthetic objects; 890+ papers reference; 12,000+ generated meshes
nuScenes	Dataset for autonomous driving, includes 3D data from sensors
SUN RGB-D	Dataset for RGB-D indoor scene understanding, contains depth images and corresponding RGB images
Stanford 3D Indoor Scene Dataset (S3DIS)	Dataset for indoor scene segmentation, includes labeled 3D point clouds of indoor spaces
Berkeley Segmentation Dataset 500 (BSDS500)	Dataset for contour detection and hierarchical segmentation
PointNet	Processes point clouds without preprocessing; uses local and global features for training
RS-CNN	Adapts 2D neural network operations; learns geometric topology restrictions between points
LDGCNN	Uses local and global processing to identify specific characteristics and relationships between structures
CurveNet	Groups point clouds by curves present in objects using undirected graphs
SimpleView	Classifies point clouds through multi-view depth images from different positions
PAConv	Uses MLPs for local portion training of point clouds, storing weights at each iteration for memory efficiency
GBNet	Applies CNN-based structures to capture geometric features of point clouds
3DMedPT	Incorporates contextual information into point clouds, initially proposed for disease detection (e.g., aneurysms)
PVT	Captures local information from non-empty voxels and refines structures in a global context using rigid object transformations

Based on different data representations, semantic segmentation and 3D classification methods can be divided into three categories: image-based *multiview*, voxel-based and point-based [25]. Our method falls into the point-based category, using the raw point cloud without any point clustering or concatenation technique with images. Classifying a dataset of real-world 3D objects remains a challenge, even with the results obtained when dealing with data generated by computer-aided design (CAD), as in the case of the ModalNet40 dataset. This is due to the complexity of the shapes and structures of real objects, which are not always completely captured by CAD models.

In the literature, some datasets stand out, for example, ShapeNet [16], ModelNet40, nuScenes [5], SUN RGB-D [26], Stanford 3D Indoor Scene Dataset (S3DIS) [27], Berkeley Segmentation Dataset 500 (BSDS500) [28], as shown in Table 1. In common, they all have more than 200 *papers* using their data, and each has more than 6 *benchmarks*. In addition to the acquisition method, the difference between them is the number of classes, the types of classes, the number of object points, and environments that were acquired, among others. In this research, we used the ModelNet40 dataset to compare our results with those of a widely recognized benchmark. ModelNet40, referenced in over 890 published articles, encompasses a diverse array of object categories and contains more than 12,000 generated meshes, providing substantial resources for comprehensive evaluation.

When analyzing benchmark results, it is essential to discuss that classification models trained on synthetic data often do not generalize well to real-world data, such as point clouds reconstructed from RGB-D scans [29, 30], and vice versa. Additionally, partial observations of real-world objects due to occlusions and sensor errors are common [31]. Furthermore, synthetic data point clouds are typically closed because they are generated mainly by design and computer-aided drafting (CAD). These factors highlight the topic's relevance, especially in the context of real-world applications, and propose significant contributions to the field's evolution. In the literature, various techniques are employed to classify point clouds. For example, PointNet [32] processes the point cloud without any pre-processing and can be trained using both local and global features.RS-CNN [33] adapts the operation of a 2D neural network and learns through the restriction of geometric topology between points. LDGCNN [34] uses two types of processing: a local one to identify specific characteristics and a global one to identify relationships between local structures. CurveNet [35] groups point clouds by types of curves present in objects (path) using undirected graphs. SimpleView [36] classifies the point cloud through multi-view; depth images generated at different positions of the original cloud. PAConv [37] uses MLPs for training on local portions of the point cloud, and the weights are stored at each iteration, resulting in savings in computational memory. GBNet [38] applies CNN-based structures to capture geometric features of point clouds. 3DMedPT [39] insertion of contextual information on point clouds, initially proposed to assist in detecting diseases, such as aneurysms. PVT [40] captures local information from non-empty voxels and refines structures in a global context using rigid object transformations.

One point cloud dataset is acquired in a real environment using a LiDAR sensor, while the other will be created in a simulated environment. This dataset is a crucial input for this work. Our analysis extends the most advanced Convolutional Neural Network (CNN) architectures, focusing on three-dimensional object recognition. We will compare different CNN configurations to determine the best approach. To evaluate neural network performance, we will use the proposed dataset as well as the ModelNet40 dataset as a reference. An important step in the process involves proposing preprocessing techniques to normalize and align the three-dimensional point clouds in the 3D Cartesian plane. The objective is to improve the quality of the CNN input data. Furthermore, we will perform a comparative analysis, examining the CNNs' results with and without the preprocessing step. We use standard evaluation metrics such as Accuracy, Precision, Sensitivity, F1 Score, and Specificity, as well as processing time and statistical tests to validate our findings. Finally, we will investigate the performance of the configured networks when fed with real data and evaluate the impact of point cloud density on 3D object classification. This project aims to provide valuable insights to improve the ability to recognize 3D objects based on point clouds.

2 Proposed Approach

This section discusses the methodology for developing the proposed approach. The subsection 2.1 illustrates, in general, the construction flow of the proposed 3D object classification method. Then, the subsection 2.2 details the configurations of the proposed architecture *Lidar3DNetV2*. The subsection 2.3 details the datasets constructed, exposing their form of acquisition and their structuring based on works in the literature. And finally, the 2.4 subsection explores the preprocessing steps covered.

2.1 3D Object Classification Method Flow

Figure 1 illustrates the flow of the method adopted for classifying 3D objects. The first stage, as shown in Figure 1, represents the acquisition and organization of the *dataset* of proposed point clouds acquired in a real environment. In the second stage, as shown in Figure 1, the point cloud normalization and transformation phases explored, which make up the pre-processing tasks proposed for the point clouds in the input objects of the *Lidar3DNetV2* network. The *Lidar3DNetV2* network topology employs the Tensorflow version 1.14 and Keras libraries for implementation. The network is built from these libraries using an input, intermediate, and output layer.

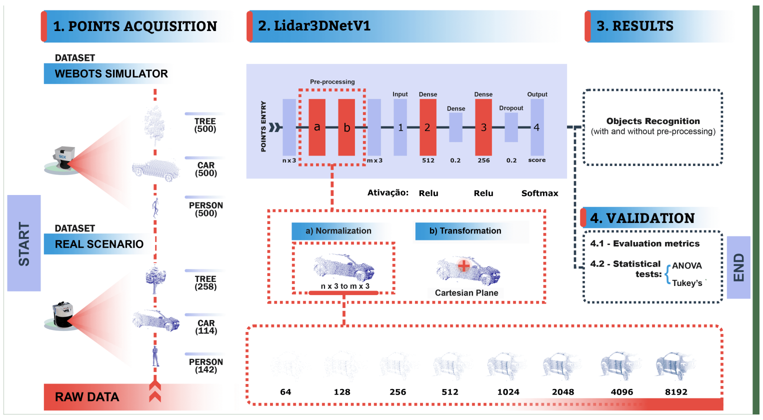

Figure 1: Flow of The Proposed Methodology.

Then, we define the dense layers using 512 hidden neurons and 256 hidden neurons in the second layer, as detailed in Figure 1.The 20% dropout is used to reduce overfitting in training, disabling part of the neurons in the intermediate layers. The activation function used was Relu in the intermediate layers and softmax in the output layer. Still in the second stage, observed in Figure 1, the variation in the number of points of the object and its respective visualization is illustrated after the normalization phase. In the third step, the output results of the proposed network *Lidar3DNetV2*, considered with and without pre-processing of point clouds, are generated. These results are discussed in the fourth stage, based on the metrics accuracy, precision, F1-Score, sensitivity, and specificity. The prediction time per sample is also considered. Still, in the fourth stage, analyses are carried out between the results with and without pre-processing point clouds from the execution of statistical tests, using the ANOVA and Turkey's methods.

2.2 Lidar3DNetV2

The Lidar3DNetV2 network classifies point clouds using an architecture based on Multilayer Perceptron (MLP). Unlike the first version [41], which used a greater number of dense layers and neurons and lacked a preprocessing step, Lidar3DNetV2 incorporates a different layer configuration and includes preprocessing. Unlike state-of-the-art artificial neural networks that use methods based on convolutional neural networks, it uses a series of operations and functions for classification and thus require high processing power. This approach with MLP proposes a less complex architecture. Therefore, it reduces the processing time of point cloud classification while maintaining relevant hit rates, contributing to good performance in applications in embedded systems.

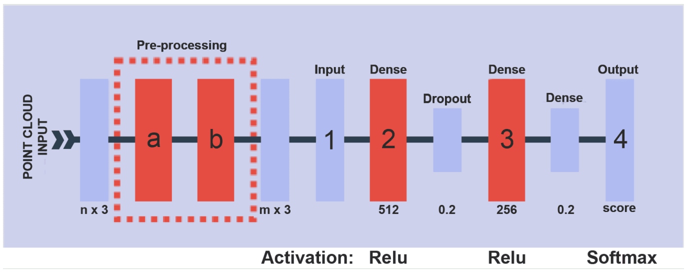

Figure 2: Lidar3DNetV2 Network Architecture.

We used Tensorflow library version 1.14 and Keras to implement the Lidar3DNet topology. Using this library, we built a network consisting of an input layer, a middle layer, and an output layer. In addition, we defined the dense layers using 512 hidden neurons and 256 hidden neurons in the second layer. Then, we used a 20% dropout to reduce overfitting in training, disabling part of the neurons in the intermediate layers. The activation function used was ReLU in the middle

The raw 3D object point cloud contains three columns with coordinate values x, y, and z. Each line in the point cloud file is a point coordinate. The first step of the Lidar3DNetV2 network architecture is to normalize the object's point clouds to 2048 points and then center the point cloud to the origin. Thus, we flattened the point clouds so that all the points information is present in a single vector. Suppose a 3D R object is represented by [[x_1, y_1, z_1], [x_2, y_2, z_2] ... [x_n, y_n, z_n]], where the point coordinate represents the vector from the point. After flattened, the vector R is represented by a single vector, as follows by [x_1, y_1, z_1, x_2, y_2, z_2 ... x_n, y_n, z_n], where n is the line number of the coordinate of the point. After the flattening step, we identified the number of points and applied an interpolation based on the nearest neighbor. In this way, it is possible to approximate the number of 3D object points and thus reach the number of 2048 points. After normalizing to 2048 points, we adjusted the 3D point cloud object to the origin of the 3D Cartesian plane through the transformation below.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t^x \\ 0 & 1 & 0 & t^y \\ 0 & 0 & 1 & t^z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1}$$

where x', y' and z' are the coordinates of the 3D object points adjusted to the origin, t is the measure of translation on the axes. After the flattening, normalization and origin adjustment methods, the point cloud of each 3D object is inserted with the Lidar3DNetV2 network.

The architecture of an MLP capable of solving non-linear problems forms the basis for the Lidar3DNetV2 network. Thus, each neuron output, performed according to the layer, can be expressed as shown in Equation 2,

$$\hat{y}_{l_k} = \phi_{l_k}\left(\sum_{i=1}^{D} \hat{y}_{l-1} w_{l_{k_i}} + b_{l_k}\right) \tag{2}$$

where l is the layer number, in this case, two layers, and k is the neuron position in that layer. Additionally, x represents the attribute vector of each point cloud, with a dimension of p. Equation 3 mathematically represents the model weights adjustment at each iteration to reduce the 3D object class error prediction.

$$\hat{w}_{l_{k_i}}(t+1) = w_{l_{k_i}}(t) + \eta \cdot \delta_{(l+1)_i}(t) \cdot \hat{y}_{l_k}(t) \tag{3}$$

where the local gradient ($\delta_{l_k}(t)$) in the l-th layer of the k-th neuron and can be expressed by the Equation 4:

$$\delta_{l_k} = \phi'_{l_k}[u_{l_k}(t)] \sum_{j=1}^{J} w_{l_{k_i}}(t) - \delta_{(l+1)_j}(t) \tag{4}$$

where z is the output layer of the network with three neurons, and ϕ' is the activation function used. The intermediate layers with 512 and 256 neurons, the activation function ReLU, expressed in the equation 5, is used. The activation function in the output layer is softmax, described in the equation 6.

$$f(x) = \max(0, x) \tag{5}$$

$$\sigma(\mathbf{z}) = \frac{e^{z_i}}{\sum_{j'=1}^{K} e^{z_j}} \tag{6}$$

where $i = 1, \ldots, K$ and $z_i = (z_1, \ldots, z_k) \in \mathbb{R}^K$, where K the number of classes of the problem, in the tests there are three classes of the proposed dataset and 40 classes in the case of dataset ModelNet40 and z_i is the output of the neuron i of the last layer of the network.

2.3 Dataset Description

There are some point cloud datasets available in the literature. For example, ModelNet [16], ShapeNet [15], ScanNet [29] and Semantic3D [42]. The dataset blueused in this work is formed by cloud point objects acquired from a MAS based on the *Light Detection and Ranging* (LIDAR) sensor. The real data were acquired in a real environment through a MAS, represented in Figure 3. The dataset generation is one major contribution of this work. We carried out the acquisitions in the external area of the Federal University of Ceará on the Fortaleza campus and in the Laboratory for Processing Images, Signals and Computer Science (LAPISCO) internal environment at the Federal Institute of Education, Science, and Technology of Ceará, campus Fortaleza.

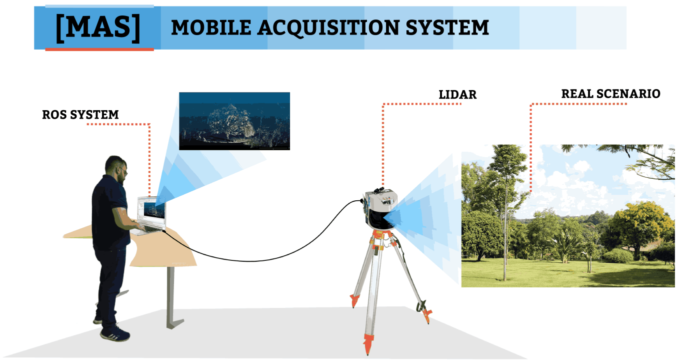

Figure 3: Mobile Acquisition System (MAS) Structure.

Unlike many existing datasets generated from CAD models or simulated environments, our dataset consists of point clouds acquired in real-world settings using a LiDAR sensor. This approach provides greater relevance and applicability for practical scenarios and it includes classes common in urban environments, such as trees, cars, and people in various poses, which are crucial for applications in vehicle navigation and free space detection.

Constructing a dataset contributes to the scientific development by acquiring points using real objects. Table 2 presents the sample distribution of the proposed dataset. The real data has a total of 484 samples, divided into different samples: 258 trees, 114 cars and 142 people in different poses. In addition to the relevant number of samples per class, there is variation in the point cloud shapes within classes. Figure 4 displays sample images and showcases the differences in the shapes of real point clouds. The car class, for example, offers three perspectives: side, front, and rear. This variation in the point cloud format can be seen in the two proposed dataset. Finally, object bridges segmentation in the raw point clouds generated by the MAS was performed manually using software CloudCompare.

The data variation significantly contributes to the excellent classifier performance in real situations, for example, in the navigation of vehicles on streets and avenues within the city. We chose classes car, tree, and person by obeying the criteria of being everyday objects in cities, specifically on roads and highways.

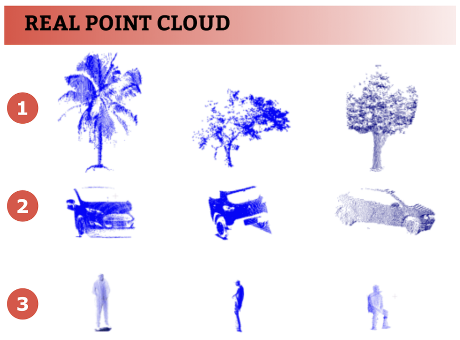

Figure 4: Example of Samples from each Class of the Dataset: (1) Tree, (2) Car, and (3) Person.

The dataset ModalNet40 is a benchmark in the literature that contains point clouds of synthetic objects. Formed by 12311 meshes generated by CAD, divided into 9843 for training and 2468 for tests, ModelNet has 40 classes of 3D objects and is widely used to analyze point clouds. Moreover, the networks chosen in this research were selected because they are benchmarks of the Modelnet40 [16]. Hence, this study chose this dataset. In addition, its point clouds are highly accurate because they are formed from CAD models.

Because it has a CAD origin, its objects are synthetic and closed, so a neural network trained with its data may have limitations when applied to classify 3D objects acquired in real environments, since they are not closed 3D objects and the distribution of points is disordered.

2.3.1 Data Acquisition

The LiDAR LMS511 is a 2D sensor that generates a distance vector of the points on the surfaces where its light beam reflects. Thus, it was necessary to develop an algorithm integrated with the ROS software to carry out the scan at a sequence of different depths and create a scene three-dimensional profile in which the sensor is inserted. Therefore, it was possible to generate the cloud points by fusing a series of distance vectors. Real data was acquired using a MAS, which has as its main component the model LMS511 LiDAR sensor, batteries, LiDAR support (tripod), notebook, and inertial measurement unit (UMI). Acquisitions were performed indoors and outdoors; we only acquired the person class in both environments.

Table 2: Sample Number per Class

Class	Dataset Real
Car	114
People	142
Tree	258

The distance between the MAS and the acquired object varies between 5 and 30 meters. This distance variation respects the range indicated by the LIDAR sensor manufacturer, which guarantees an operating range of up to 80 meters [43].On the other hand, the variation in distance interferes with the number of points and the distribution of points in the generated point cloud, challenges expected because they are generated in a real environment. Also, in the person and car point clouds acquisition, the variation in distance ranged between 5 and 10 meters. In the tree class, the acquisition distance alternated from 10 to 30 meters, due to the size of the trees. The data distribution, number of samples present in each class and the division according to the type of data collection is presented in Table 2. The dataset built with the real-world acquisitions is named LLR514 (LAPISCO-LiDAR real-world dataset), while the dataset comprising the synthetic point clouds is named LLS1500 (LAPISCO-LIDAR synthetic dataset).

After acquiring and labeling the point clouds, the dataset was shuffled and divided into three equal parts to perform cross-validation, a method that involves training and testing a machine learning model on different clusters of the same dataset. Each model was trained and tested n times, where n is the number of partitions, using n-1 partitions for training and the remaining partition for testing in each iteration. This allows evaluating the 3D Object Classification models on different data arrangements, resulting in n sets of evaluation metrics. During the three cross-validation iterations, all networks were trained in 50 epochs, maintaining the original settings, except for changing the number of neurons in the output layers from 40 to 3 classes, according to the number of classes in the proposed dataset.

2.4 Pre-processing Cloud Point Approach

Pre-processing methods on data in signals aim to extract attributes, standardize or parameterize data to improve the next stage process of classification, segmentation or visualization [44]. The following sections present the techniques applied to the proposed point cloud data. This step is one of the contributions of this work.

2.4.1 Point Normalization

The normalization of the object's point cloud is a crucial preprocessing step in the proposed method and a key contribution of this work. Many neural networks employed for object classification and recognition from point clouds, such as PointNet, utilize a fixed input size of 2048 points, for instance [32] [45].Given that the number of points in an object's point cloud can vary based on its size, for example, a tree's point cloud typically has more points than that of a person, the technique of normalizing the number of points aims to standardize the point count in the clouds while preserving their unique characteristics and structure. In this approach, the method of upsampling is used to add synthetic samples and downsampling is used to remove them as needed. [46].

The inter nearest method is applied to point clouds based on the nearest neighbor concept for trilinear interpolation [47] and thus resize the object's point cloud by combining eight different points, which implies the less loss of information in the resulting set of points. Considering that a point cloud initially has P points and at the end of this step the cloud of points is normalized in N points, for example 2048 points. The first step for this situation to be successful and not modify or reduce the shape of the cloud of points is to find the proportion of points between the dimensions of each Cartesian axis. Through the Equation 7 we have the proportions.

$$s_X = s_Y = s_Z = \frac{P}{N} \tag{7}$$

Knowing that s_X is the ratio for the x axis, for the y axis this ratio is s_Y and s_Z is the ratio for the z axis. In this way, it is possible to calculate the new position of a point on each axes, through Equation 8,

$$x_f = y_f = z_f = x' s_X = y' s_Y = z' s_Z, \ \forall \ x = y = z = 1, ..., N \tag{8}$$

so we have the change in x given by $\Delta_x = x_f - x$, the change in y described by $\Delta_y = y_f - y$ and the change in z being $\Delta_z = z_f - z$.

This way, the pre-processed cloud points are calculated by combining eight points that form a cube, expressed by (x, y, z), $(x+1, y, z)$, $(x, y+1, z)$, $(x, y, z+1)$, $(x+1, y, z+1)$, $(x, y+1, z+1)$, $(x+1, y+1, z)$ and $(x+1, y+1, z+1)$. And the equation 9 represents this calculation.

$$\begin{aligned} K(x', y', z') = {} & L(x,y,z)(1-\Delta_x)(1-\Delta_y)(1-\Delta_z) + L(x+1,y,z)\Delta_x(1-\Delta_y)(1-\Delta_z) + \\ & L(x,y+1,z)(1-\Delta_x)\Delta_y(1-\Delta_z) + L(x,y,z+1)(1-\Delta_x)(1-\Delta_y)\Delta_z + \\ & L(x+1,y,z+1)\Delta_x(1-\Delta_y)\Delta_z + L(x,y+1,z+1)(1-\Delta_x)\Delta_y\Delta_z + \\ & L(x+1,y+1,z)\Delta_x\Delta_y(1-\Delta_z) + L(x+1,y+1,z+1)\Delta_x\Delta_y\Delta_z \end{aligned} \tag{9}$$

where K is the preprocessing result and L is the original point cloud.

Figure 5 illustrates the normalization technique applied to the three classes of the proposed dataset. Also, one can see the point cloud classes normalization, car and person of 9961, 11189 and 2316 points, respectively, in 2048 points. These samples are present in the dataset developed in this research. The distance between the LiDAR sensor and the object interferes with the number of points and concentration of points in certain regions of the object's point clouds. However, the pre-processing step, specifically normalization, reduces this limitation in the acquisition of point clouds.

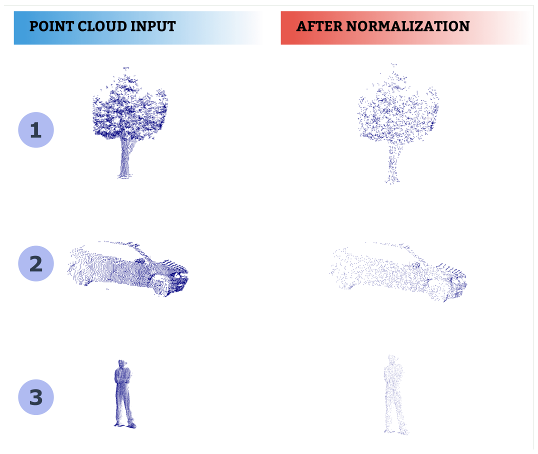

Figure 5: Point Cloud Normalization in each Dataset Class. Adapted from [41].

The images demonstrate that the added or eliminated points do not interfere, incisively, in the object shape because the manipulated points are very close to each other. And, in certain sections of the point cloud, it may appear that two or more points overlap, even though they have distinct coordinates. This phenomenon arises from the way point size is configured during the figure's generation. The point can be reduced when generating the figure, but the image visualization quality reduces and prevents a good visualization.

2.4.2 Cartesian Plan Adjustments

Following the normalization step, the preprocessing focuses on aligning the 3D object point cloud with the Cartesian plane. During this phase, a transformation is applied to the points in 3D Euclidean space. The translation operation relocates the point cloud of the object to the origin of the 3D Cartesian plane. It is crucial to emphasize that during this preprocessing step, the point cloud remains unchanged; no points are added, removed, or distorted, and the default origin is preserved for all object point clouds. Equation 1 can mathematically express the transformation operation. Approaches that use the point cloud origin knowledge strategy are also found in [48, 49].

Figure 6 illustrates the preprocessing step for origin adjustment applied to objects with different spatial formats. The point cloud of a sphere and a cube were used to exemplify this step. The origin of the point cloud is highlighted in all figures with a plus symbol "+" and shaded. The origin in the point clouds is generated at the acquisition time, so the positions are varied. This adjustment stage in the Cartesian plane is one of the research contributions and caused a direct interference in the best performance of the neural networks. The results of this step will be discussed in Section 3.

The table validating metrics with the ANOVA [50] statistical test is represented in the table 7 and 9. The ANOVA statistical method evaluates whether the classification methods differ or are statistically equivalent, that is, whether or not the choice of network matters for classification purposes. For this test we consider the following hypotheses:

- *H_0* - All networks are statistically equivalent. Which means that there is no difference between using one network or another for the 3D Object Classification task;

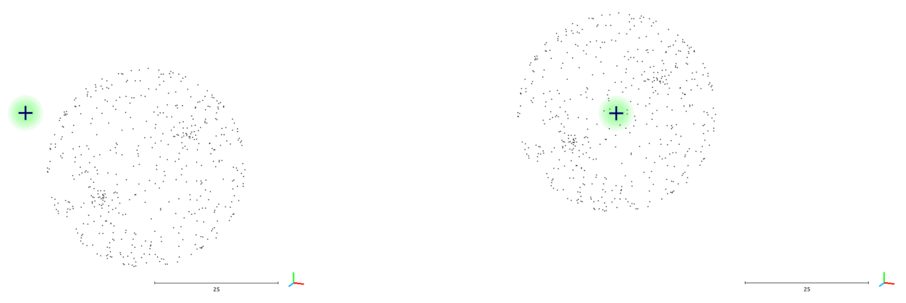
(a) Origin outside the center of the point cloud. (b) Origin at the center of the point cloud.

Figure 6: Point cloud adjustment step in the 3D Cartesian plane for origin.

- H_1 - At least one network is different from another. Which means that there is a network with better or worse results than the others.

According to the tables 7 and 9, p is less than 0.05, so we can reject the null hypothesis H_0, considering that at least one network differs from another when compared, implying that there is one better than the other. The statistical test applies to all evaluation metrics because in all cases, p is less than 0.05. However, the ANOVA method does not highlight the best or worst method. Thus, the Tukey [51] test is applied to statistically quantify how different neural network classification methods are, considering the hypotheses below:

- H_0 - The evaluated network is equivalent to the other network in comparison. Which means there is no difference between using one network or another for the 3D Object Classification task;

- H_1 - Networks are different from each other. Therefore, there is a difference in static significance between using one or another network for the 3D object classification task.

The same reference value p less than 0.05 is considered to differ. However, if p is more significant than 0.05, it is possible to statistically quantify how many classification methods are equivalent, which can reach a maximum of 1. In this case study, the maximum is 0.9 because the ANOVA test indicates that all classification methods have a difference. We performed all statistical tests for both ANOVA and Tukey approaches using the *bioinfokit* library at version 1.0.3.

3 Results and Discussion

This section presents the results, discussions and comparisons between the state-of-the-art neural networks and the proposed network Lidar3DNetV2, considering the methodology and evaluation metrics described in section 2. All environments were configured in hardware with Ubuntu operating system, processor Intel Core i7, 8 GB of RAM, video card GeForce GTX 1070. In addition, libraries such as Python 3.x, scikit-learn, Pytorch, Tensorflow and Keras have been used with different versions. In table 3, the tools and versions of the models used were highlighted. For this reason, the docker technology feature was applied and each neural network has a specific container.

The first subsection 3.1 discusses the configuration validation of state-of-the-art network environments. In the subsection 3.2, real, original and pre-processed data are applied to the input of the neural networks. In addition to discussions about Anova and Turkey static tests.

3.1 Validation of Network Configurations using ModelNet40 Dataset

In the initial step, all networks were set up following the guidelines provided in the respective repositories as specified by each research work to commence testing.To validate the configurations made, we executed the architectures using the ModelNet40 dataset and adhered to the training and testing instructions provided by the original authors. Table 4 presents a comparison between the accuracy results reported by the authors and the accuracy results achieved in our configured architectures, including the Lidar3DNetV2 network proposed in this study. All selected networks were replicated in a Docker environment respecting the authors' configurations. All compared networks underwent training and testing with the Modelnet40 and LLR514 datasets (LAPISCO-LiDAR real-world dataset).

Equality and, in some cases, approximation between the accuracies of the networks is expected because the configuration and dataset are similar to those published by the authors in their studies. Processing time was not reported due to the difference in computational resources between the machines on which the tests were performed. However, it is something that does not interfere with the objective of this first step, which is to validate the network configuration. The specific result of the Lidar3DNetV2 network did not obtain a relevant result when compared to the others. However, as specified in Section 2, the Lidar3DNetV2

Table 3: Libraries and their Versions used in each Model.

Model	Configuration
Lidar3DNetV2	tensorflow-gpu: 1.14.0 keras: 2.2.3 Python: 3.7
3DMedPT	Python: 3.7 Pytorch: 1.6 CUDA: 10.0 Packages: tqdm, sklearn, visualdl, einops, natsort
CurveNet	Python: $\geq$ 3.7 PyTorch: $\geq$ 1.2 Packages: glob, h5py, sklearn
SimpleView	Python version: 3.7.5 CUDA version: 10.0 CuDNN version: 7.6 GCC version: 5.4
GBNet	Python: 3.6 Pytorch: 0.4.0 CUDA: 9.1
RSCNN	Ubuntu: 16.04 Python: 3.6 (recommended Anaconda3) Pytorch: 1.0+ (testado em 1.0, 1.1 e 1.2) CMake: $>$ 2.8 CUDA: 10.2 cuDNN: 7.6 GNU: $\leq$ 7.5
PVT	Python: $\geq$ 3.7 PyTorch: $\geq$ 1.3 tensorboardX: $\geq$ 1.2
POINTNET	TensorFlow: $\geq$ 1.4 (GPU version) Python: 2.7 CUDA: 8.0 cuDNN: 5.1
PAConv	PyTorch: $\geq$ 1.5.0 Python: $\geq$ 3.0 CUDA: $\geq$ 10.1 tensorboardX
LDGCNN	tensorflow: 1.4.0 Python: 3.5.2 CUDA: 8.0.61 cuDNN: 6.0.21

Table 4: Validation of the architecture with Accuracy (Acc) metrics, in percentage (%), of each network on the ModelNet40 dataset.

Network	*Accuracy(%)*	
	Original Paper	**Replication**
3DMedPT	93.40	93.20
Lidar3DNetV2	80.63	80.70
CurveNet	94.20	94.20
GBNet	91.04	91.04
LDGCNN	92.90	92.90
PAConv	93.90	93.90
PointNet	89.20	89.20
PVT	94.00	94.00
RSCNN	93.60	93.60
SimpleView	93.90	93.90

network is designed based on the proposed real dataset which is formed by three object classes and the ModelNet40 dataset has 40 different object classes. Even though the Lidar3DNetV2 network is designed for real data and three objects its accuracy of 80.63% after training and testing with the Modelnet40 data is satisfactory.

Other works, for example, DeepPano [52] and 3DShapeNets [16] also used the dataset ModelNet40 and had accuracy below 80%, for this reason they were not added in Table 4 and not considered in the research of this work. The SimpleView network achieved one of the top results. However, it is important to note that in its configuration, the point clouds of the objects consist of 1024 points. This aspect should be taken into consideration in the subsequent discussions.

3.2 Evaluation using the Real Dataset as Input in State-of-The-Art Networks with Original and Pre-Processed Data

With the configurations validated, the next step is to run the configured networks by changing the networks' input data, replacing the ModelNet40 data with *datasets* with real data proposed in this work. The Tables 5 e 6 present the performance of state-of-the-art neural networks and the proposed Lidar3DNetV2 network, with original data and with data applied to the pre-processing stage, respectively. The tables are composed of seven columns that highlight the neural network under analysis, the subsets created in the cross-validation stage and the following five columns contain the evaluation metrics. In tables 5 and 6 the best accuracy of each neural network, considering all subsets, are highlighted in blue.

The RSCNN network obtained significant results in accuracy. When applying real data to the RSCNN input, accuracy reached 100% in both configurations. Because RSCNN uses the CNN architecture as a base and extends it to irregular point cloud analysis configurations, the proposed pre-processing step did not significantly interfere with the evaluation metrics.

RSCNN adapted to the original and pre-processed data as the network learns through the geometric topology restriction

Table 5: Metrics, in percentage (%), of each Neural Network and each test subset over the original and real. Considering all subsets, the best accuracy results of each network were highlighted in blue.

Metric \ Network	3DMedPT	Lidar3DNetV2	CurveNet	GBNet	LDGCNN	PAConv	PointNet	PVT	RSCNN	SimpleView	
Accuracy	58.72	80.23	89.53	40.70	91.25	**84,30**	**87.79**	**93.60**	97.09	86.63	Subset 1
F1-Score	47.54	77.13	87.39	29.42	89.27	80.61	86.12	92.63	97.09	86.63	
Precision	55.31	77.75	87.36	26.21	88.41	81.16	85.67	92.16	96.35	84.64	
Sensibility	54.86	76.65	87.51	33.88	90.67	80.78	86.90	93.16	97.27	88.08	
Specificity	54.86	89.54	87.51	33.88	90.67	80.78	86.90	93.16	97.27	88.08	
Accuracy	59.41	**84.80**	90.06	37.43	92.46	81.87	82.74	90.64	98.83	87.72	Subset 2
F1-Score	56.18	83.06	88.61	34.07	91.32	77.22	81.73	90.09	98.82	87.81	
Precision	64,92	82.22	87.77	34.51	90.41	79.06	80.88	88.94	98.92	87.21	
Sensibility	61.02	84,87	89.83	35.78	92.66	78.04	84.10	91.84	98.74	91.05	
Specificity	61.02	93.00	89.83	35.78	92.66	78.04	84.10	91.84	98.74	91.15	
Accuracy	**75,29**	75.44	**90.64**	**50.29**	**92.50**	78.94	84.52	91.22	**100.00**	**90.06**	Subset 3
F1-Scores	73.04	71.24	89.49	22.31	91.58	72.89	82.06	90.16	**100.00**	89.30	
Precision	73.49	71.47	89.45	16.76	91.39	74.45	81.47	89.19	**100.00**	88.75	
Sensibility	75.32	71.69	89.57	33.33	91,91	74.31	83.20	91.75	**100.00**	91.15	
Specificity	75.32	87.84	89.57	33.33	91.91	74.31	83.20	91.75	**100.00**	91.15	

Table 6: Metrics, in percentage (%), of each Neural Network and each test subset over the pre-processed and real. Considering all subsets, the best accuracy results of each network were highlighted in blue.

Metric \ Network	3DMedPT	Lidar3DNetV2	CurveNet	GBNet	LDGCNN	PAConv	PointNet	PVT	RSCNN	SimpleView	
Accuracy	69.18	97.13	93.60	34.88	95.62	97.79	81.39	95.93	98.25	94.76	Subset 1_c
F1-Scores	51.77	95.74	92.90	30.17	95.18	86.23	77.14	95.70	97.99	94.03	
Precision	49.33	94.95	92.47	31.32	94.35	87.61	79.20	94.76	97.56	93.42	
Sensibility	60.85	95.59	93.47	31.35	96.14	86.96	78.47	96.98	98.53	94.73	
Specificity	60.85	98.71	93.47	31.35	96.14	86.96	78.47	96.98	98.53	94.73	
Accuracy	70.58	**97.70**	90.56	36.25	95.00	**88.88**	86.66	94.73	98.24	95.90	Subset 2_c
F1-Scores	57.71	96.83	92.53	31.65	94.67	87.60	76.05	94.16	98.26	95.64	
Precision	75.39	95.61	92.29	31.63	93.71	86.87	77.05	93.53	98.52	95.87	
Sensibility	63.63	97.87	92.81	32.26	95.87	89.20	81.86	94.88	98.02	95.49	
Specificity	63.73	96.98	92.81	32.28	95.87	89.20	81.64	94.76	98.18	95.39	
Accuracy	**73.52**	96.85	**95.90**	**38.59**	96.82	87.13	**92.85**	**97.66**	**100.00**	**98.83**	Subset 3_c
F1-Scores	66.97	92.19	95.36	36.51	96.51	84.47	92.14	97.50	**100.00**	98.40	
Precision	68.18	93.42	95.82	36.69	96.35	86.83	91.97	97.12	**100.00**	98.63	
Sensibility	67.70	92.53	95.32	36.75	96.82	85.10	92.71	97.96	**100.00**	98.24	
Specificity	67.46	92.84	95.30	36.62	96.94	85.12	92.75	97.93	**100.00**	98.25	

between points, where real data contributes to good performance. Considering the processing time for classifying each sample, illustrated in Figures 7 and 8, RSCNN requires a lot of time compared to the others, ranking among the three slowest.

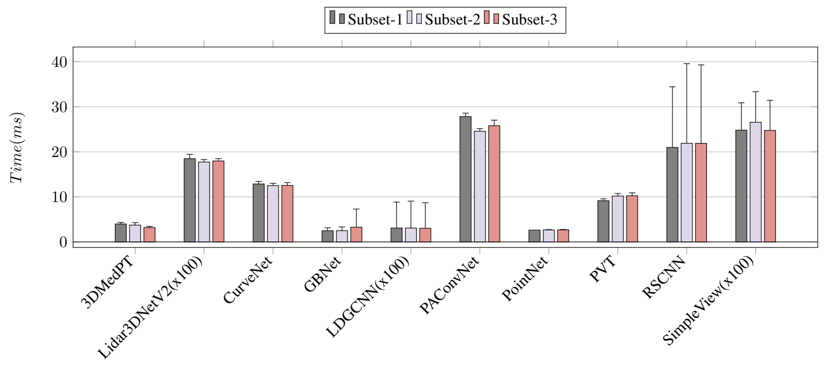

Figure 7: Average time and standard deviation to execute the predictions on each sample of the point clouds, in milliseconds, for all networks in the experiment on the original and real data

.

In *dataset benchmark* ModelNet40 the PVT network is between the two networks with accuracy above 94%. When the proposed data was inserted into the input of the PVT network, the accuracy reduced approximately 1% in The best set and additional metrics exceeded 92%, demonstrating the method's capability to accurately identify the object's class. It effectively indicates which objects do not belong to the class and correctly classifies 100% of the positive samples. Its configuration based on attempts to identify empty voxels and refine structures with rigid transformations are factors that affect the network's performance.

With architecture designed to classify point clouds from medical images, expressly point clouds from closed objects, such as blood vessels. The 3DMedPT neural network reduced values in the accuracy metric, approximately 18%, when the ModelNet40 input data was replaced by *dataset* with real point clouds. Its good performance on ModelNet40 reached accuracy above 93% because the point clouds are generated in CAD and are also closed clouds. When applying the pre-processed point clouds, the values did not change significantly because the method was based on the point clouds' transformation. Therefore, it is evident

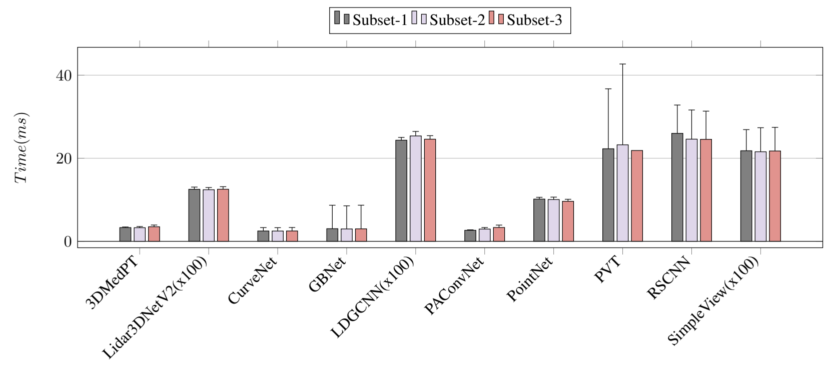

Figure 8: Average time and standard deviation to execute predictions on each point cloud sample, in milliseconds, for all experiment networks on pre-processed and real.

that the proposed preprocessing method is straightforward yet effective in classifying the object accurately.

Considering that the PointNet neural network uses 2048 points as input for 3D objects, a comparison of the accuracy results between Table 4 and Table 5 showed close and expected values. However, upon introducing the dataset with real point clouds, the accuracy decreased by 2%, which might be due to the reduction in the number of classes. Considering that the PointNet neural network uses direct data at its input, like the Lidar3DNetV2 network, the normalization, and rendering of point clouds considerably improve its performance.

The results of the CuverNet network, with the point clouds of the proposed dataset, the accuracy reduced by approximately 4% in relation to the ModelNet40 result. Knowing that its method of guiding the path when going through the points was trained with point clouds drawn in software (CAD), when applying real point clouds, which do not have an ordered distribution of points equal to those generated in CAD, the reduction in the efficiency of classification expected. After the point clouds were pre-processed, the CurveNet network result recovered almost 5%, in cluster 3. With the point clouds adjusted to the same origin, the path orientation performed by the method can have the same origin as a reference , contributing to better performance.

The decrease in accuracy metrics from 93.90% with the ModelNet40 dataset to 90% in the best subset of our proposed real dataset and 86% in the worst subset can be attributed to the increase in the number of object points, which was raised to 2048 points. This increase was intended to create a level playing field for all networks. However, it is worth mentioning that the SimpleView network is originally trained with 1024 points, and this divergence likely impacted its performance, leading to reduced metrics.However, after the proposed pre-processing step, the SimpleView network obtained one of the best increases in the percentage of its metrics.

The PAConv network had a negative highlight because its accuracy reduced by 15% in its lowest result, compared to the 93.90% of ModelNet40. However, when applied to the real pre-processed data, the metrics reached about 89% but were not higher than those of the ModelNet40 benchmark. Considering that the PAConv network is based on kernel construction, the use of real and unordered data impairs its performance. Applied to the MobileNet40 network, GBNet reached 93% accuracy, but when inserting data with real point clouds, its accuracy reduced to 50% at its best performance. Some factors contribute to this result, such as the application of real data, the number of points at the entrance to the network, as the author used 1024 for his tests, and the low resource of geometric information in certain regions of the cloud of real points, as the clouds of points real data do not have a regular distribution of points. With the point cloud pre-processing step, GBNet reduced its performance even more and stood out with the worst performance, with accuracy and other metrics below 38%. Redundancy in the generation of local characteristics of the point cloud is a relevant factor for low performance, even with pre-processed data that in other networks provide a gain in performance.

Table 7: One-way ANOVA test results for experiments on original and real data.

Metric		**Df**	**SS**	**MS**	**F-*value***	**p-*value***
Accuracy	C(Networks)	9,00	0,72	0,08	45,85	0,00
	Residual	20,00	0,03	0,00	-	-
F1-*Score*	C(Networks)	9,00	1,13	0,12	46,75	0,00
	Residual	20,00	0,05	0,00	-	-
Precision	C(Networks)	9,00	1,14	0,12	56,84	0,00
	Residual	20,00	0,04	0,00	-	-
Sensibility	C(Networks)	9,00	0,96	0,10	59,74	0,00
	Residual	20,00	0,03	0,00	-	-
Specificity	C(Networks)	9,00	0,23	0,02	82,91	0,00
	Residual	20,00	0,00	0,00	-	-

The results of the Tukey HSD statistical test are presented in Tables 8 and 10. The tables are formed by seven columns; the first one is the reference subset, in this case, the network that will be confronted individually with the others. Thirty-six matches were held. Underneath the reference network is its accuracy value for easy viewing when comparing networks.

In the second column is the subset2, in which all the networks that were not confronted with the reference network are

Table 8: Tukey HSD test results on original and real data. Values in bold text mean that the compared networks are statistically distinct from each other p-*value*.

Subset	**Subset2**	**Accuracy**	**F1-*Score***	**Precision**	**Sensibility**	**Specificity**
3DMedPT *(Acc = 75,29%)*	Lidar3DNetV2	0,005	0,010	0,017	0,016	0,008
	CurveNet	0,001	0,001	0,001	0,001	0,001
	GBNet	0,001	0,001	0,001	0,001	0,001
	LDGCNN	0,001	0,001	0,001	0,001	0,001
	PAConv	0,007	0,005	0,002	0,03	0,002
	PointNet	0,001	0,001	0,004	0,003	0,001
	PVT	0,001	0,001	0,001	0,001	0,001
	RSCNN	0,001	0,001	0,001	0,001	0,001
	SimpleView	0,001	0,001	0,001	0,001	0,001
Lidar3DNetV2 *(Acc = 84,80%)*	CurveNet	**0.090**	**0.109**	**0.112**	**0.158**	**0.170**
	GBNet	0.001	0.001	0.001	0.001	0.001
	LDGCNN	**0.391**	**0.342**	**0.412**	**0.474**	**0.505**
	PAConv	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	PointNet	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	PVT	**0.091**	**0.072**	**0.062**	**0.058**	**0.070**
	RSCNN	0.0041	0.036	0.046	0.046	0.045
	SimpleView	**0.792**	**0.734**	**0.785**	**0.786**	**0.853**
CurveNet *(Acc = 90,64%)*	GBNet	0,001	0,001	0,001	0,001	0,001
	LDGCNN	**0,900**	**0,900**	**0,900**	**0,900**	**0,900**
	PAConv	**0,356**	**0,226**	**0,287**	**0,086**	**0,199**
	PointNet	**0,880**	**0,900**	**0,900**	**0,900**	**0,872**
	PVT	**0,900**	**0,900**	**0,900**	**0,900**	**0,900**
	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	**0,900**	**0,900**	**0,900**	**0,900**	**0,900**
GBNet *(Acc = 50,29%)*	LDGCNN	0,001	0,001	0,001	0,001	0,001
	PAConv	0,001	0,001	0,001	0,001	0,001
	PointNet	0,001	0,001	0,001	0,001	0,001
	PVT	0,001	0,001	0,001	0,001	0,001
	RSCNN	0,001	0,001	0,001	0,001	0,001
	SimpleView	0,001	0,001	0,001	0,001	0,001
LDGCNN *(Acc = 92,50%)*	PAConv	**0,135**	**0,087**	**0,125**	0,016	0,046
	PointNet	**0,564**	**0,731**	**0,642**	**0,579**	**0,456**
	PVT	**0,900**	**0,900**	**0,900**	**0,900**	**0,900**
	RSCNN	**0,644**	**0,683**	**0,508**	**0,593**	**0,528**
	SimpleView	**0,900**	**0,900**	**0,900**	**0,900**	**0,900**
PAConv *(Acc = 84,30%)*	PointNet	**0,600**	**0,704**	**0,688**	**0,590**	**0,688**
	PVT	**0,156**	**0,079**	**0,124**	0,012	0,047
	RSCNN	**0,054**	**0,054**	**0,056**	**0,056**	**0,056**
	SimpleView	**0,665**	**0,316**	**0,465**	0,045	0,307
PointNet *(Acc = 87,79%)*	PVT	**0,605**	**0,702**	**0,638**	**0,499**	0,459
	RSCNN	0,020	0,045	0,016	0,017	0,008
	SimpleView	**0,900**	**0,900**	**0,900**	0,838	**0,900**
PVT *(Acc = 93,60%)*	RSCNN	**0,604**	**0,713**	**0,511**	**0,672**	**0,526**
	SimpleView	**0,900**	**0,900**	**0,900**	**0,900**	**0,900**
RSCNN *(Acc = 100,00%)*	SimpleView	**0,126**	**0,289**	**0,144**	**0,332**	**0,105**

present. And the other columns are the evaluation metrics adopted in the research. The first highlight is the GBNet and 3DMedPT networks that differ from each other and from the others. With an accuracy of 75.29%, 3DMedPT did not obtain values close to the best results and was also far from the GBNet network, which obtained 50.29%. The Lidar3DNetV2 network, with 84.80%, had an accuracy value close to the CurveNet, LDGCNN, PAConv, PointNet, PVT and SimpleView networks and thus the value of *p* was highlighted in black in the Table 8. The RSCCNN network had a good performance and reached 100% accuracy in the classification of 3D objects and when confronted with networks with accuracy above 88%, they showed a minimum statistical equivalence. The LGDCNN, CurveNet, PVT, PointNet and SimpleView networks achieved accuracy close to 90% and reached a maximum value of 0.9 when confronted.

The Table 10 was modified in comparison to Table 8 due to the inclusion of the preprocessing step for the networks, most of them, improved their performance and the networks Lidar3DNetV2, PointNet and PAConv reached percentages above 92%, thus increasing the statistical similarity with the other networks that presented this accuracy. In this way, the highlighted p-*value* number increased and many with 0.900. The performance gain, validated by the evaluation metrics and statistical tests, it is possible to highlight that the proposed pre-processing step is relevant for the task of classifying 3D objects based on point clouds, using real data.

However, in order to carefully assess which classification method stands out, the average time to perform the prediction of each sample was calculated and is illustrated in Figures 7 and 8. The RSCNN network, achieving 100% in all metrics, was one of the three networks requiring the most time to predict the object class, along with PVT and PointNet. The times of the Lidar3DNetV2, LDGCNN and SimpleView networks were multiplied by 100 (x100) due to their processing time being well below the others and when plotting the graph the visualization would be impaired. Thus, the networks with the lowest average time to perform the prediction of the object class, thus being the most suitable for applications involving navigation of autonomous vehicles and in real time. Considering the accuracy analysis together with the average prediction time analysis, the Lidar3DNetV2 network obtained the best performance and thus, for real data, it is the most suitable method to classify 3D objects based on point clouds.

Table 9: Results of the *one-way* ANOVA test on pre-processed (S_c) and real data.

Metric		**Df**	**SS**	**MS**	**F-*value***	**p-*value***
Accuracy	C(nets)	8.00	0.96	0.12	56.53	3.49e-11
	Residual	18.00	0.03	0.00	-	-
F1-*Score*	C(nets)	8.00	1.20	0.15	32.72	3.47e-09
	Residual	18.00	0.08	0.00	-	-
Precision	C(nets)	8.00	1.10	0.13	23.99	4.32e-08
	Residual	18.00	0.10	0.00	-	-
Sensibility	C(nets)	8.00	1.12	0.14	46.13	1.96e-10
	Residual	18.00	0.05	0.00	-	-
Specificity	C(nets)	8.00	0.26	0.03	77.02	2.43e-12
	Residual	18.00	0.00	0.00	-	-

Table 10: Tukey test results on real, original and pre-processed data (S_c). Values in bold text mean that the compared networks are statistically distinct from each other p-*value*.

Subset1	**Subset2**	**Accuracy**	**F1-*Score***	**Precision**	**Sensibility**	**Specificity**
3DMedPT *(Acc = 73,52%)*	Lidar3DNetV2	0.001	0.001	0.001	0.004	0.001
	CurveNet	0.001	0.001	0.001	0.004	0.001
	GBNet	0.001	0.001	0.004	0.002	0.001
	LDGCNN	0.001	0.001	0.001	0.002	0.001
	PAConv	0.001	0.003	0.001	0.009	0.001
	PointNet	0.025	0.030	0.053	0.051	0.010
	PVT	0.001	0.001	0.001	0.001	0.001
	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	0.001	0.001	0.001	0.001	0.001
Lidar3DNetV2 *(Acc = 97,70%)*	CurveNet	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	GBNet	0.001	0.001	0.001	0.001	0.001
	LDGCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	PAConv	**0.363**	**0.390**	**0.301**	**0.353**	**0.321**
	PointNet	**0.603**	**0.604**	**0.703**	**0.703**	**0.690**
	PVT	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	RSCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
CurveNet *(Acc = 95,90%)*	GBNet	0.001	0.001	0.001	0.001	0.001
	LDGCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	PAConv	**0.699**	**0.812**	**0.899**	**0.900**	**0.854**
	PointNet	**0.722**	**0.712**	**0.759**	**0.780**	**0.754**
	PVT	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	RSCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	0.900
GBNet *(Acc = 38,59%)*	LDGCNN	0.001	0.001	0.001	0.001	0.001
	PAConv	0.001	0.001	0.001	0.001	0.001
	PointNet	0.001	0.001	0.001	0.001	0.001
	PVT	0.001	0.001	0.001	0.001	0.001
	RSCNN	0.001	0.001	0.001	0.001	0.001
	SimpleView	0.001	0.001	0.001	0.001	0.001
LDGCNN *(Acc = 96,82%)*	PAConv	**0.458**	**0.547**	**0.690**	**0.900**	**0.538**
	PointNet	**0.304**	**0.210**	**0.212**	**0.252**	**0.246**
	PVT	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	RSCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
PAConv *(Acc = 88,88%)*	PointNet	**0.502**	**0.605**	**0.505**	**0.596**	**0.500**
	PVT	**0.411**	**0.486**	**0.652**	**0.900**	**0.495**
	RSCNN	**0.010**	**0.015**	**0.031**	**0.037**	**0.029**
	SimpleView	**0.346**	**0.526**	**0.626**	**0.820**	**0.554**
PointNet *(Acc = 92,85%)*	PVT	**0.583**	**0.540**	**0.510**	**0.545**	**0.530**
	RSCNN	**0.292**	**0.342**	**0.492**	**0.306**	**0.443**
	SimpleView	**0.472**	**0.407**	**0.490**	**0.432**	**0.460**
PVT *(Acc = 97,66%)*	RSCNN	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
RSCNN *(Acc = 100,00%)*	SimpleView	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**

4. Conclusion

This work proposes a multilayer perceptron-based architecture for classifying 3D objects based on point clouds. The model presents higher metrics than different models in different databases, being statistically different, and for the models that present better performance, there is no statistical difference from ours. Furthermore, the model propose is an MLP neural network architecture without convolutional layers, which implies a smaller number of parameters, lower memory, and disk consumption, and is computationally less expensive, both for training and inference. In this way, the model presents itself as a possible embedded solution for classifying 3D objects in point clouds in indoor and outdoor environments. The study selects the main state-of-the-art neural networks and compares them, using five evaluation metrics and two statistical evaluation methods, with the proposed network, Lidar3DNetV2, using dataset with real data.

The discussion stemming from the data variation in the proposed architecture makes a valuable contribution to the scientific community's efforts in classifying point clouds. This contribution is especially significant as it involves the evaluation of various state-of-the-art networks using both realdata. Additionally, it encompasses a range of preprocessing steps and includes testing with five different metrics as well as two statistical tests for thorough assessment.

Initially, ten distinct environments, each designed for a specific artificial neural network, were successfully configured. The input for these environments was the ModelNet40 dataset. Notably, the Curvenet and PVT networks achieved accuracy levels surpassing 94%. Subsequently, these configured networks were fed with data from the proposed dataset, which includes point clouds of real objects and incorporates preprocessing. The test results reveal a significant disparity in network performance when real data are applied. All networks, except 3DNetPT, demonstrated gains of over 5% in the evaluation metrics. Notably, the Lidar3DNet network exhibited remarkable improvement, soaring from 84.80% to 97.70% accuracy.

Lidar3DNet not only delivered exceptional accuracy but also boasted one of the shortest prediction times per sample, alongside the LDGCNN network. However, the proposed Lidar3DNet network outperformed others in terms of accuracy in tests with real data. Therefore, it stands as the most suitable choice for embedded system applications, considering the five evaluation metrics and classification processing time. For future work, our plans include developing a third version of Lidar3DNet and incorporating 3D object segmentation into our study. Lastly, we aim to evaluate the suitability of our models for applications in embedded systems, particularly in the context of the Internet of Things and Edge Computing.

REFERENCES

[1] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen and S.-K. Yeung. "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data". In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1588–1597, Seoul,KOR, 2019. IEEE.

[2] L. Guo and P. Wang. "Art product design and VR user experience based on IoT technology and visualization system". *Journal of Sensors*, vol. 2021, 2021.

[3] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*. "Scalability in perception for autonomous driving: Waymo open dataset". In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2446–2454, Seattle,DC, 2020. IEEE.

[4] J. Mei, A. Z. Zhu, X. Yan, H. Yan, S. Qiao, L.-C. Chen and H. Kretzschmar. "Waymo open dataset: Panoramic video panoptic segmentation". In *European Conference on Computer Vision*, pp. 53–72, Dan Pnorama Hotel, Tel Aviv, 2022. Springer.

[5] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan and O. Beijbom. "nuscenes: A multimodal dataset for autonomous driving". In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, Long Beach,CA, 2020. IEEE Xplore.

[6] R. Fan, H. Wang, P. Cai and M. Liu. "Sne-roadseg: Incorporating surface normal information into semantic segmentation for accurate freespace detection". In *European Conference on Computer Vision*, pp. 340–356, Honolulu,HI. CVPR.

[7] I. Nagy and F. Oniga. "Free Space Detection from Lidar Data Based on Semantic Segmentation". In *2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 95–100, Cluj-Napoca,RO, 2021. IEEE.

[8] J. Fan, M. J. Bocus, B. Hosking, R. Wu, Y. Liu, S. Vityazev and R. Fan. "Multi-scale feature fusion: Learning better semantic segmentation for road pothole detection". In *2021 IEEE International Conference on Autonomous Systems (ICAS)*, pp. 1–5, Montréal,CA, 2021. IEEE, IEE.

[9] Z. Chen, J. Zhang and D. Tao. "Progressive lidar adaptation for road detection". *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 693–702, 2019.

[10] P. G. Ravishankar, A. M. Lopez and G. M. Sanchez. "Unstructured Road Segmentation using Hypercolumn based Random Forests of Local experts". 2022.

[11] H. Wang, R. Fan, Y. Sun and M. Liu. “Applying surface normal information in drivable area and road anomaly detection for ground mobile robots”. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2706–2711, online, 2020. IEEE.

[12] T.-H. Chen and T. S. Chang. “RangeSeg: range-aware real time segmentation of 3D LiDAR point clouds”. *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 1, pp. 93–101, 2021.

[13] S. Mohapatra, S. Yogamani, H. Gotzig, S. Milz and P. Mader. “BEVDetNet: bird’s eye view LiDAR point cloud based real-time 3D object detection for autonomous driving”. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 2809–2815, Indianapolis,IN, 2021. IEEE.

[14] Y. A. Alnaggar, M. Afifi, K. Amer and M. ElHelw. “Multi projection fusion for real-time semantic segmentation of 3d lidar point clouds”. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1800–1809, Waikoloa,HI, 2021. IEEE.

[15] 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”. 2015.

[16] 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*, pp. 1912–1920, Boston, MA, 2015. IEEE.

[17] J. Li, B. M. Chen and G. Hee Lee. “So-net: Self-organizing network for point cloud analysis”. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9397–9406, Seattle,WA, 2018. CVPR.

[18] L. Tchapmi, C. Choy, I. Armeni, J. Gwak and S. Savarese. “Segcloud: Semantic segmentation of 3d point clouds”. In *2017 international conference on 3D vision (3DV)*, pp. 537–547, Qingdao,CN, 2017. IEEE.

[19] L. Landrieu and M. Simonovsky. “Large-scale point cloud semantic segmentation with superpoint graphs”. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4558–4567, San Juan,PR, 2018. IEEE.

[20] A. Kotb, S. Hassan and H. Hassan. “A Comparative Study Among Various Algorithms for Lossless Airborne LiDAR Data Compression”. In *2018 14th International Computer Engineering Conference (ICENCO)*, pp. 17–21, Cairo,EG, 2018. IEEE.

[21] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam and P. Vandergheynst. “Geometric deep learning: going beyond euclidean data”. *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[22] Y. Wu, L. Liu, C. Pu, W. Cao, S. Sahin, W. Wei and Q. Zhang. “A comparative measurement study of deep learning as a service framework”. *IEEE Transactions on Services Computing*, 2019.

[23] L. Cao, I. Goreshnik, B. Coventry, J. B. Case, L. Miller, L. Kozodoy, R. E. Chen, L. Carter, A. C. Walls, Y.-J. Park *et al.*. “De novo design of picomolar SARS-CoV-2 miniprotein inhibitors”. *Science*, vol. 370, no. 6515, pp. 426–431, 2020.

[24] A. Holzinger, G. Langs, H. Denk, K. Zatloukal and H. Müller. “Causability and explainability of artificial intelligence in medicine”. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 4, pp. e1312, 2019.

[25] Z. Hu, M. Zhen, X. Bai, H. Fu and C.-l. Tai. “Jsenet: Joint semantic segmentation and edge detection network for 3d point clouds”. In *European Conference on Computer Vision*, pp. 222–239, online, 2020. Springer.

[26] S. Song, S. P. Lichtenberg and J. Xiao. “Sun rgb-d: A rgb-d scene understanding benchmark suite”. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 567–576, Boston, MA, 2015. IEEE.

[27] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer and S. Savarese. “3d semantic parsing of large-scale indoor spaces”. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas,NV. IEEE.

[28] P. Arbelaez, M. Maire, C. Fowlkes and J. Malik. “Contour detection and hierarchical image segmentation”. *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 5, pp. 898–916, 2010.

[29] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser and M. Nießner. “Scannet: Richly-annotated 3d reconstructions of indoor scenes”. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5828–5839, Honolulu,HI, 2017. CVPR.

[30] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu and S.-K. Yeung. “Scenenn: A scene meshes dataset with annotations”. In *2016 fourth international conference on 3D vision (3DV)*, pp. 92–101, Stanford,CA, 2016. Ieee.

[31] S. Song and J. Xiao. “Deep sliding shapes for amodal 3d object detection in rgb-d images”. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 808–816, Las Vegas, NV, 2016. IEEE.

[32] C. R. Qi, H. Su, K. Mo and L. J. Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, Honolulu,HI, 2017. IEEE/CVF.

[33] Y. Liu, B. Fan, S. Xiang and C. Pan. "Relation-Shape Convolutional Neural Network for Point Cloud Analysis", 2019.

[34] K. Zhang, M. Hao, J. Wang, X. Chen, Y. Leng, C. W. de Silva and C. Fu. "Linked Dynamic Graph CNN: Learning through Point Cloud by Linking Hierarchical Features". In *2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. IEEE, November 2021.

[35] T. Xiang, C. Zhang, Y. Song, J. Yu and W. Cai. "Walk in the Cloud: Learning Curves for Point Clouds Shape Analysis", 2021.

[36] A. Goyal, H. Law, B. Liu, A. Newell and J. Deng. "Revisiting Point Cloud Shape Classification with a Simple and Effective Baseline", 2021.

[37] M. Xu, R. Ding, H. Zhao and X. Qi. "PAConv: Position Adaptive Convolution with Dynamic Kernel Assembling on Point Clouds", 2021.

[38] S. Qiu, S. Anwar and N. Barnes. "Geometric Back-projection Network for Point Cloud Classification", 2021.

[39] J. Yu, C. Zhang, H. Wang, D. Zhang, Y. Song, T. Xiang, D. Liu and W. Cai. "3D Medical Point Transformer: Introducing Convolution to Attention Networks for Medical Point Cloud Analysis", 2021.

[40] C. Zhang, H. Wan, X. Shen and Z. Wu. "PVT: Point-Voxel Transformer for Point Cloud Learning", 2022.

[41] P. H. F. de Sousa, J. S. Almeida, E. F. Ohata, F. G. Nogueira, B. C. Torrico, V. H. C. de Albuquerque, M. M. Hassan, N. Kumar, M. R. Hassan and P. P. Rebouças Filho. "Intelligent 3d objects classification for vehicular ad hoc network based on lidar and deep learning approaches". *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[42] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler and M. Pollefeys. "Semantic3d. net: A new large-scale point cloud classification benchmark". 2017.

[43] P. Amaradi, N. Sriramoju, L. Dang, G. S. Tewolde and J. Kwon. "Lane following and obstacle detection techniques in autonomous driving vehicles". In *2016 IEEE International Conference on Electro Information Technology (EIT)*, pp. 0674–0679, Grand Forks,ND, 2016. IEEE.

[44] S. Hafeez and N. Kathirisetty. "Effects and Comparison of different Data pre-processing techniques and ML and deep learning models for sentiment analysis: SVM, KNN, PCA with SVM and CNN". In *2022 First International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR)*, pp. 1–6, Hyderabad,IN, 2022. IEEE.

[45] Y. Aoki, H. Goforth, R. A. Srivatsan and S. Lucey. "Pointnetlk: Robust & efficient point cloud registration using pointnet". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7163–7172, Long Beach,CA, 2019. IEEE Xplore.

[46] R. l. Toral and A. Chakrabarti. "Generation of Gaussian distributed random numbers by using a numerical inversion method". *Computer physics communications*, vol. 74, no. 3, pp. 327–334, 1993.

[47] P. Bourke. "Interpolation methods". *Miscellaneous: projection, modelling, rendering*, vol. 1, no. 10, 1999.

[48] Y. Zhou and O. Tuzel. "Voxelnet: End-to-end learning for point cloud based 3d object detection". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4490–4499, Salt Lake City, UT, 2018. IEEE.

[49] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang and O. Beijbom. "Pointpillars: Fast encoders for object detection from point clouds". In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12697–12705, Long Beach,CA, 2019. IEEE.

[50] E. R. Girden. *ANOVA: Repeated measures*. Number 84. Sage, 1992.

[51] H. Abdi and L. J. Williams. "Tukey's honestly significant difference (HSD) test". *Encyclopedia of research design*, vol. 3, no. 1, pp. 1–5, 2010.

[52] B. Shi, S. Bai, Z. Zhou and X. Bai. "Deeppano: Deep panoramic representation for 3-d shape recognition". *IEEE Signal Processing Letters*, vol. 22, no. 12, pp. 2339–2343, 2015.