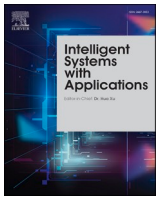
Intelligent Systems with Applications 17 (2023) 200162

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications



Robust pooling through the data mode

Ayman Mukhaimar [*,a], Ruwan Tennakoon [b], Reza Hoseinnezhad [a], Chow Yin Lai [c], Alireza Bab-Hadiashar [a]

[a] school of engineering, RMIT, 124 La Trobe St, Melbourne, 3000, VIC, Australia
[b] school of science, RMIT, 124 La Trobe St, Melbourne, 3000, VIC, Australia
[c] UCL, Gower Street, London, United Kingdom

ARTICLE INFO

Keywords:
Robust classification
Point cloud
Robust segmentation
Robust noise estimation

ABSTRACT

The task of learning from point cloud data is always challenging due to the often occurrence of noise and outliers in the data. Such data inaccuracies can significantly influence the performance of state-of-the-art deep learning networks and their ability to classify or segment objects. While there are some robust deep-learning approaches, they are computationally too expensive for real-time applications. This paper proposes a deep learning solution that includes novel robust pooling layers which greatly enhance network robustness and perform significantly faster than state-of-the-art approaches. The proposed pooling layers replace conventional pooling layers in networks with global pooling operations such as PointNet and DGCNN. The proposed pooling layers look for data mode/cluster using two methods, RANSAC, and histogram, as clusters are indicative of models. We tested the proposed pooling layers on several tasks such as classification, part segmentation, and points normal vector estimation. The results show excellent robustness to high levels of data corruption with less computational requirements as compared to robust state-of-the-art methods. our code can be found at https://github.com/AymanMukh/ModePooling.

1. Introduction

The use of deep learning for several 3D task such as point cloud classification (Esteves et al., 2018; Klokov and Lempitsky, 2017; Qi et al., 2017a; Ramasinghe et al., 2019; Su et al., 2015; Wu et al., 2015a), retrieval (Ramasinghe et al., 2019; Wu et al., 2015a), and segmentation (Qi et al., 2017a; 2017b; Wang et al., 2019) has shown great success in recent years. However, the success has largely been confined to 3D CAD-based benchmarks such as ModelNet (Wu et al., 2015a), McGill (Siddiqi et al., 2008), and Shapenet (Chang et al., 2015b) with very clean data. Working with 3D point clouds of real scenes where data are inaccurate and may be corrupted by outliers remains a challenge, and real 3D training data for natural and man-made objects are still scarce. Testing recent deep networks on 3D CAD models perturbed with outliers and noise showed that the data perturbation has a huge influence on the classification performance (Mukhaimar et al., 2019a). One approach to resolve this issue is to train the network with outliers. However, noise and outliers, by definition, are not predictable and it would be difficult to train the network for all possible scenarios. Another approach is to build a robust framework that can diminish the influence of outliers compared to conventional deep networks (Gould et al., 2019). This approach has received significant interest in recent years (Gould et al., 2019; Mukhaimar et al., 2019b; 2022).

To demonstrate the effect of data perturbation on a deep neural network performance, we show the effect of the existence of outliers on PointNet (Qi et al., 2017a) in Fig. 1. PointNet consists of several layers of multilayer perceptrons (MLP), a max pooling layer, fully connected layers (FC), and a classification layer. We show in Fig. 1C the distribution of one of the feature vectors (with a dimension of $1 \times N$, where $N = 2048$) before the pooling operation when the network is trying to classify: (1) the point cloud of a chair, and (2) the same chair corrupted with 50% outliers. When the max pooling is used, the output of the pooling layer in the case of the clean chair is 2.98, while the output of the pooling layer when outliers exist is 6.46. This huge difference affects the classification accuracy as seen in Fig. 1D. Interestingly, the figure also shows that using the mean or the median of the data does not diminish the effect of outliers. As such, a better pooling operation is needed to diminish the effect of data inaccuracies on 3D data processing.

The use of M-estimators in Deep Declarative Network (Gould et al., 2019) implementation showed promising results for perturbed data.

* Corresponding author.
E-mail address: ruwan.tennakoon@rmit.edu.au (R. Tennakoon).

https://doi.org/10.1016/j.iswa.2022.200162
Received 27 July 2022; Received in revised form 2 November 2022; Accepted 27 November 2022
Available online 1 December 2022

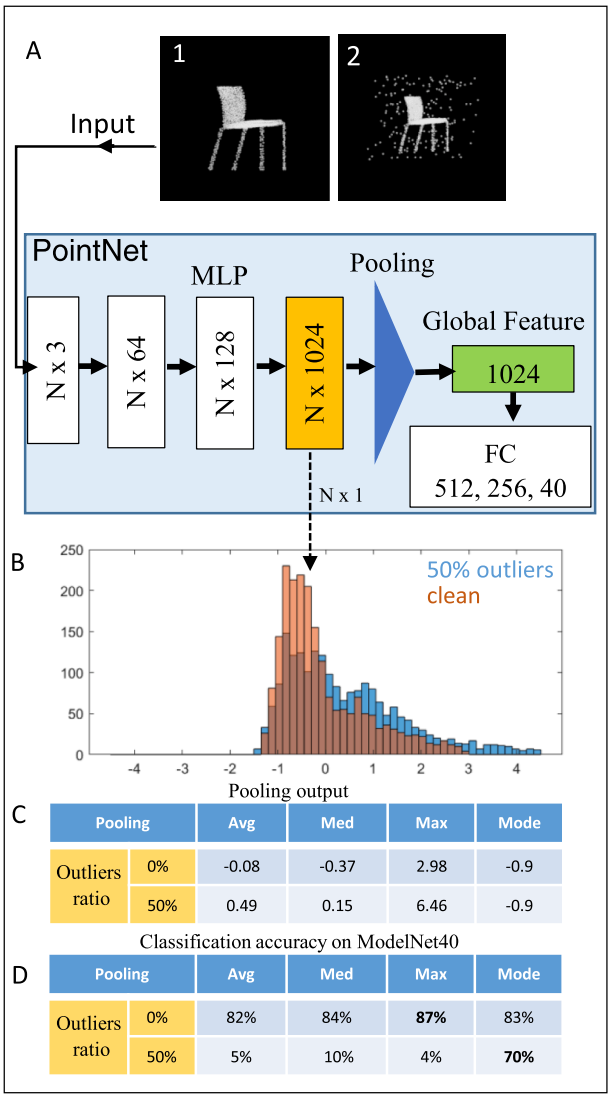


Fig. 1. (A) PointNet vanilla classification architecture. We show two scenarios for object classification: (1) The classification of the point cloud of a chair, and (2) The classification of the point cloud of the same chair that is corrupted with 50% outliers. (B) Histogram of one of the feature vectors before the pooling operation. The orange histogram shows the data distribution for the clean chair, while the blue histogram shows the distribution for the chair with the corrupted point cloud. (C) The output of the pooling layer in PointNet(shown as the blue triangle) in case we have average pooling, median pooling, max pooling, and Mode pooling. (D) The classification accuracy for the selected pooling layers.

Both Truncated quadratic (TQ) or WELSCH (W) when used in the pooling operation achieved significantly better robustness to outliers. M-estimators look for data mode which is robust to outliers as seen in Fig. 1C & D. However, M-estimators have several shortcomings such as the high computation requirements, and the solution is either non-convex, non-smooth or not very robust. Thus, inspired by the use of M-estimators, we investigate other alternatives that also look for data mode and achieve robust pooling.

In this paper, we propose two robust pooling layers. The first one uses RANSAC (Fischler and Bolles, 1981) framework for finding the location of the mode, while the other one uses a histogram-based pooling method. Both RANSAC and histogram methods look for high cluster regions and should give similar results to M-estimators while being computationally cheaper. Moreover, unlike many M-estimators, both RANSAC and histogram provide unique solutions and despite RANSAC being an iterative solution, our results show that it can still be faster than using M-estimators. However, RANSAC computational requirements in still expensive and grows exponentially with the size of the data. On the other hand, our proposed histogram-based pooling layer is shown to be significantly faster than the above methods.

The histogram pooling layer divides the feature data into uniform regions and selects regions with maximum densities. In theory, the histogram is similar to both RANSAC and M-estimators, where bin size is somewhat equivalent to the inlier/outlier threshold of RANSAC or tuning parameters of M-estimators. But unlike both approaches, using the histogram mode is significantly cheaper and enables the network to be used for real-time applications. The testing time was found to be around 150 times faster than both RANSAC or an M-estimator, and the computational complexity was found to be similar to conventional pooling methods such as max pooling.

Unlike existing robust approaches that are limited to object classification, e.g. Mukhaimar et al. (2019b), Riegler et al. (2017), Mukhaimar et al. (2022), the proposed framework is able to also perform robust classification, segmentation, and points normal estimation. We conducted an extensive set of experiments on both clean and perturbed data for object classification, segmentation, and points normal estimation, showing that the proposed methods have high classification accuracy and compete in robustness with state-of-art methods. We summarize the contributions of this paper as follows:

- Two novel pooling layers for point cloud classification, segmentation, and normal's estimation are presented. The pooling layers are robust against point cloud noise perturbations and other types of data corruptions such as outliers.
- Compared to other robust methods, the proposed pooling layers are significantly cheaper in computation and enable the neural network-based methods to be used for real-time applications.
- The proposed pooling layers can also be used in neural networks with a global pooling layer such as PointNet, and DGCNN (Wang et al., 2019).

The rest of this paper is structured as follows: We first discuss the latest deep learning classification networks. We then explain the intuition behind using data mode instead of maximum or average pooling as well as the inner working of the proposed RANSAC and histogram-based pooling frameworks. This is followed by the analysis of the performance of the proposed pooling layers under different data corruption in Section 4. Section 5 presents a sensitivity analysis of network parameters, followed by Section 7, which concludes the paper.

2. Related work

Recent deep learning frameworks for 3D point cloud classification can be categorized as Multi-view CNNs (MV-CNN) (Shi et al., 2015; Su et al., 2015), Voxel-based CNNs (Wu et al., 2015b; Xiang et al., 2019; Zhou and Tuzel, 2018), and point-based CNNs (Chen et al., 2019; Qi et al., 2017a; 2017b; Zhang et al., 2019b). The robustness of a number of these networks, from the different categories, was examined extensively (Mukhaimar et al., 2019a). The study compared PointNet, PointNet++ (Qi et al., 2017b), Kd-Net (Klokov and Lempitsky, 2017), Oct-Net (Riegler et al., 2017), and MV-CNN (Su et al., 2015) for data with outliers, noise, and missing points. The study showed that the classification performance of MV-CNN and PointNet++ was heavily affected by the above data perturbations and corruption forms. As MV-CNN uses several 2D images of a 3D model (i.e. 70 images captured from different locations), outliers would appear in most images, and the overall number of outliers is magnified by the imaging process (70 times). PointNet showed good performance against noise and missing points but was heavily affected by outliers. Both PoinNet and PointNet++ use max pooling as part of their networks, which causes the networks to select outliers as maximum values during their pooling operations. The Kd-Net was heavily affected by outliers as the existence of outliers changes the

structure of the Kd-tree graph that performs the classification. Oct-Net showed a good level of robustness to noise and outliers but was affected by missing points. We used the same testing framework and tested other methods that are based on PoinNet and PointNet++ such as PVCNN and RS-CNN and our results showed similar performance to the original PoinNet and PointNet++ methods for the corrupted data.

In terms of robust methods that exist in the literature, Pl-Net3D (Mukhaimar et al., 2019b) is a feature-based method that combines a primitive fitting technique with PointNet to achieve robust classification. The method employs RANSAC to find instances of geometric primitives in 3D point clouds. The features of those primitives are then used to classify objects. R-SCNN (Mukhaimar et al., 2022) is a voxel-based method that uses spherical harmonics-based CNNs. The method was able to achieve state-of-art robustness but was limited to object classification. Deep declarative networks (Gould et al., 2019) proposes a framework for optimization methods to be implemented as part of deep learning networks. The proposed framework allows robust statistical approaches such as M-estimators (Leroy and Rousseeuw, 1987) to be implemented in deep learning. Both Pl-Net3D and M-estimators involve significant computation and high processing time, limiting their ability for real-time usage.

In this paper, we present a robust classification framework that competes with state-of-art methods in terms of accuracy, robustness, and computational load. The proposed approach is based on PointNet, thus object classification, segmentation, and other point-based manipulations, such as points normal estimation, are possible. The proposed approach can be also adapted to any PointNet-based methods such as DGCNN (Wang et al., 2019) or LDGCNN (Zhang et al., 2019a).

3. Method

Given a point cloud of an object that contains outliers, our objective is to build a robust classification deep learning network. To achieve robustness, we propose to use a robust pooling operation in our network. We introduced robust pooling to PointNet (Qi et al., 2017a) and DGCNN (Wang et al., 2019) architectures as both methods use global pooling operations. PointNet architecture is shown in Fig. 1A. The multi-layer perceptron (MLP) followed by the Pooling operation is commonly used in most of the point cloud deep learning networks. The architecture represents a symmetric operation on all points, which enables the network to work with unsorted inputs. DGCNN uses similar architecture except that the network accounts for neighboring points. The pooling operation is the main reason for such architecture to fail to classify objects as seen from Fig. 1, hence achieving robust pooling improves the overall robustness.

3.1. Problem statement

For the given block diagram shown in Fig. 1A, the feature mapping block (shown in orange color) provides N feature vectors in the $\mathbb{R}^{1024}$ space. In the dataset used to benchmark the performance of the proposed solution in our paper, typically there are around $N = 2048$ points in the cloud that are mapped (through a MLP) into $N = 1024$ features.

A major task embedded in the diagram shown in Fig. 1A is the *pooling* task (shown in cyan colour), where it *aggregates* the N vectors and obtain a *single vector* as the *best* representative of the information contents of those vectors. Let us denote the ith feature vector by

$$\boldsymbol{x}_i = \left(x_{i,1}, x_{i,2}, \ldots, x_{i,D}\right)$$

where $D = 1024$ and $i = 1, 2, \ldots, N$. The N features are then treated as samples of a joint feature density $p : \mathbb{R}^D \rightarrow \mathbb{R}$. The information content of the point cloud is then represented by the entropy of the density that is given by:

$$E\left[p\right] \triangleq -\int p\left(\boldsymbol{x}\right)\log\left(p\left(\boldsymbol{x}\right)\right)d\boldsymbol{x}. \tag{1}$$

Given the samples, we approximate the density as the sum of Dirac delta terms centered at the samples and weighted by their point pdf values:

$$p(\boldsymbol{x}) \approx \sum_{i=1}^{N} p(\boldsymbol{x}_i)\delta(\boldsymbol{x} - \boldsymbol{x}_i) \tag{2}$$

Substituting the above approximation in Eq. (1) returns the following approximation for entropy:

$$E[p] \approx -\sum_{i=1}^{N} p(\boldsymbol{x}_i)\log(p(\boldsymbol{x}_i)) \tag{3}$$

Among all the terms included in the summation, the largest one is associated with the maximum aposteriori (MAP) estimate of the feature, i.e.

$$\widehat{\boldsymbol{x}}_{\text{MAP}} = \underset{\boldsymbol{x}}{\text{argmax}} p(\boldsymbol{x}) \tag{4}$$

Therefore, we choose the MAP estimate as the most informative aggregate of all the N feature (the output of the pooling operation in Fig. 1A).

The main problem is how to estimate the MAP. In practice, we are have around $N = 2048$ features in a space dimension of $D = 1024$. In terms of a mesh grid, this is equivalent to having $m = \lceil \log_D N \rceil = 2$ bins per dimension in a 1024-D histogram, which is not sufficient for the purpose finding the peak of the joint density.

Our solution to this problem is to find an approximate for the peak of the very high-dimensional joint density by forming marginal densities (D instances of them, one for each dimension), and locate the peak of each marginal density, separately, then put the coordinates of those peak points together to form the approximate location of the peak of the joint density in the D-dimensional feature space. In other words, we find an estimate, named *Marginal MAP* (MMAP) estimate, given by

$$\widehat{\boldsymbol{x}}_{\text{MMAP}} = (\widehat{x}_{1_{\text{MAP}}}, \cdots, \widehat{x}_{D_{\text{MAP}}}),$$

where $\widehat{x}_{i_{\text{MAP}}}$, $i = 1, \ldots, D$, is the MAP estimate for the marginal density in the ith dimension.

In our application, assuming that the MLP and Feature Map networks in Fig. 1A are trained, we expect MMAP and MAP estimates to be reasonably close to each other in such a way that MMAP estimate can be still declared as the aggregate feature that holds a substantial amount of information encapsulated in the N feature samples produced by the Feature map block in Fig. 1A.

To explain the intuition behind the above statement, first, note that the fully-connected network that inputs the aggregated (global) feature is indeed a mapping from the 1024-D space to 40 different classes of objects. As such, the outputs of the fully-connected network are expected to be very close to one of the coordinate unit vectors $\boldsymbol{e}_j = [\boldsymbol{0}_{i-1}^{\top} \quad 1 \quad \boldsymbol{0}_{40-i}^{\top}]^{\top}$ where $\boldsymbol{0}_k$ means a k-dimensional vector of zeros. Thus, we intuitively expect that after the network is trained, the global feature input to the fully-connected network ends up in one of 40 different zones in the 1024-D space that are quite distinct. In fact, this is what is expected for the N feature samples; they end up being located together within one of those 40 zones. Hence, with no outlier samples, we expect to see a single-peak distribution of the features (similar to a multivariate Gaussian), and for such a distribution, the peak location (MAP estimate) and the MMAP estimate are very close if not identical.

An example is shown in Fig. 2, demonstrated in 2D for the purpose of visualization. Fig. 2(a) presents the density of data comprised of 80% outlier samples that are uniformly distributed and inliers being distributed according to a joint Gaussian. We observe that the peak is at $[0\ 0]^{\top}$, while the peaks of the two marginal densities shown in Fig. 2(b) are both located at zero.

If the outliers are not uniformly distributed, as long as they do not themselves form a sharper peak in the density, we still expect the MMAP estimate to be close to the peak location. This is visualized in an example shown in Fig. 3 where 25% of data are the inliers distributed in a similar way to the previous case, and the rest of the data (outliers) are equally scattered around four points. Fig. 3(b) demonstrates that due to the outliers, the marginal density peaks have slightly deviated from the peak of the combined density.

3.2. Histogram and RANSAC pooling

Robust fitting techniques aim to find data clusters that represent instances of a given model. To apply RANSAC, a collection of m hypotheses was examined to find the inliers within a threshold ϵ for all these m hypotheses. The hypothesis with the maximum number of inliers is then chosen as the best model estimate. This translates to looking for:

$$\widehat{m} = arg\,max_m\Big(\sum_{i=0}^{N} |x_i - x_m| \leq \epsilon\Big) \tag{5}$$

where $\widehat{m}$ corresponds to the point with the maximum number of inliers (equivalent to $\widehat{x}_{\mathrm{MAP}}$) and the output of the pooling layer in the forward step.

For one-dimensional data, a histogram can be viewed as a density estimator where data is partitioned into intervals (bins) and their density is estimated by counting the number of data in a bin. We use a histogram as part of our proposed pooling operation, in which $\widehat{L}$ is the index of the mode bin $\widehat{L} = arg\,max_m p(L_i)$, where $L \subset R^m$ is a set of bin indices for m bins. In comparison to RANSAC for location estimation, the bin size is equivalent to the threshold ϵ and the histogram mode is equivalent to the model with the maximum number of inliers.

4. Experiments

In this section, we present a comparative analysis of the performance of the proposed pooling operations for different types of data perturbation and corruption. We perform experiments on different tasks, including classification, part-segmentation, and points normal vector estimation. We also outline the composition of the datasets as well as the network architectures.

4.1. Datasets

For classification, we use the ModelNet40 (Qi et al., 2017a; Wu et al., 2015a), ShapeNet (Chang et al., 2015a), and ScanObjectNN (Uy et al., 2019) datasets. ModelNet40 consists of 9,843 training and 2468 testing samples from 40 categories. Each sample consists of 2048 points normalized within the unit cube. We don't introduce any augmentation to the training data except random rotations, but data perturbations and corruptions such as noise, random point dropout, and outliers, are introduced to the testing samples. Examples of those perturbations and corruptions are seen in Fig. 4. If a point normal is used, we calculate the normal by using twenty of its neighboring points. We use the ScanObjectNN dataset (Uy et al., 2019) to test the performance of our proposed pooling operations on real-scene data. The ScanObjectNN dataset contains 2902 scenes of objects categorized into 15 categories. Each scene carries the point cloud of an object in addition to the point cloud of background elements or parts of nearby objects as seen in Fig. 4. During training, we use the point cloud of objects only, while when testing, we include the point cloud of background and parts of nearby objects as real-scenarios outliers.

For part segmentation, we use ShapeNet part dataset (Yi et al., 2016), which consists of 16,881 shapes from 16 categories, with 50 parts in total. All images are annotated with their parts labels. To examine the robustness, we corrupted the test set with random outliers, and when testing, we only used inlier points to calculate the average mIoU.

For points normal estimation, we use the ModelNet40 dataset. Each object consists of 2048 points, and each point is labeled with its normal vector. We train methods without any data augmentations, while we perturb the testing dataset with different levels of noise.

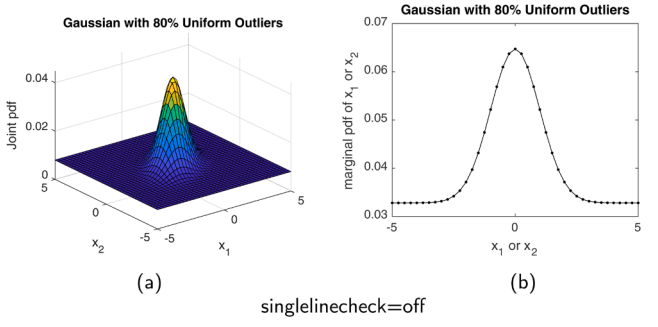


Fig. 2. **(a)** Density plot of a 2D Gaussian density $\mathcal{N}(\cdot;\boldsymbol{\mu}_1,\Sigma_1)$ with $\boldsymbol{\mu}_1 = \boldsymbol{0}$ and $\Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$ mixed with 80% outlier samples that are distributed uniformly in $[-5\ 5] \times [-5\ 5]$. **(b)** Marginal densities of x_1 and x_2 for the joint density shown in part (a) of this figure.

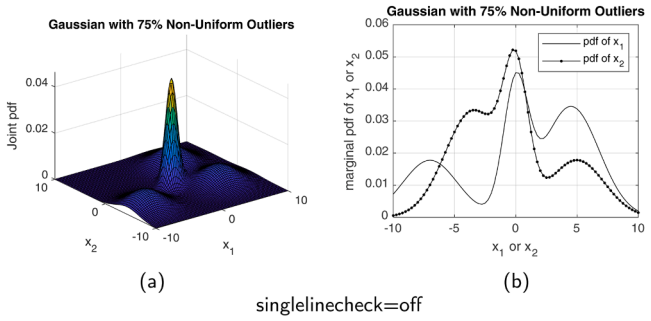


Fig. 3. **(a)** A Gaussian mixture density in 2D, comprised of four components $\mathcal{N}(\cdot;\boldsymbol{\mu}_i,\Sigma_i), i = 1,\ldots,4$ with equal weight 0.25. The first component is same as shown in Fig. 2(a). The parameters of the other components are: $\boldsymbol{\mu}_2 = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$, $\boldsymbol{\mu}_3 = \begin{bmatrix} -3 \\ 5 \end{bmatrix}$, $\boldsymbol{\mu}_3 = \begin{bmatrix} -4 \\ 7 \end{bmatrix}$, $\Sigma_2 = \Sigma_3 = \Sigma_4 = \begin{bmatrix} 5 & 0.5 \\ 0.5 & 5 \end{bmatrix}$ **(b)** Marginal densities of x_1 and x_2 for the joint density shown in part (a) of this figure.

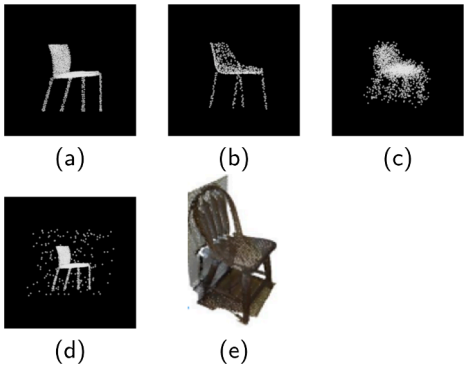


Fig. 4. **(a)** The point cloud of a chair taken from the ModelNet40 dataset, **(b)** the same chair is corrupted with random point dropout, **(c)** the same chair is perturbed with Gaussian noise, **(d)** the same chair is corrupted with scattered outliers, and **(e)** the point cloud of a chair taken from ScanObjectNN dataset including background data (used as pseudo outliers).

4.2. Selected architectures

To analyze the performance of the proposed pooling operation, we used the PointNet architecture as its multi-layer perceptrons and global pooling are shared by many recent deep learning frameworks. For classification with PointNet, three layers of MLP were used with 64,128,1024 filters respectively. The number of bins for histogram pooling was set to 70 and their centers were uniformly distributed between -10 to 10. RANSAC was implemented with an equivalent threshold of 0.143 and with the number of hypotheses m ranging between 30% to 50% of the total number of points. The learning rate was set to 0.0001 and the number of epochs was set to 100. For ScanObjectNN, two layers of MLP were used with 128 and 4048 filters respectively, and the number of bins for histogram pooling was set to 200. We also investigate DGCNN in the sensitivity analysis section (see section for detail).

For part segmentation and normals estimation, we use the original PointNet segmentation architecture with the proposed histogram pooling. The number of bins in the histogram was set to 1200 for the interval between -5 to 5. The initial learning rate was set to 0.0001, and the number of epochs was set to 100.

4.3. Classification performance on ModelNet40, ShapeNet, and ScanObjectNN datasets

In this section, we present the classification accuracy of our proposed framework against ModelNet40, ShapeNet, and ScanObjectNN datasets. We calculated the classification accuracy for the PointNet model with different pooling operations including max, RANSAC (RN), histogram (HS), and Truncated Quadratic (TQ). The results of these experiments as well as the classification accuracy for state-of-the-art methods such as PointCNN (Li et al., 2018), CurveNet (Xiang et al., 2021), VoxNet (Maturana and Scherer, 2015), PointNet++ (Qi et al., 2017b), and PL-Net3D (Mukhaimar et al., 2019b) are shown in Table 1. The classification accuracy on ModelNet40 when using "PointNet (vanilla)" with max pooling reaches 87%, while when using TQ, HS, and RN pooling operations, the classification accuracy reaches 83.7%, 83.7%, and 81.6% respectively. When points normal are used, the classification accuracy for those pooling operations increases by 2-3%. The classification accuracy on ScanObjectNN for max, TQ, HS, and RN reaches 82%, 74%, 79%, and 76% respectively. However using points normal increases the classification accuracy of TQ, HS, and RN to 83%, 82%, and 81.2% respectively. When using the ShapeNet dataset and points normal, the classification accuracy for max, TQ, HS, and RN reaches 82%, 80.3%, 79%, and 76% respectively.

As can be seen from Table 1, the performance of the proposed pooling operations is comparable to state-of-art methods such as PointCNN, CurveNet, and PointNet++. The slightly lower accuracy on clean datasets is compensated by the robustness to different data corruptions and perturbations as will be shown next.

Table 2 shows the classification accuracy when training was performed on ModelNet40 and when testing was done on ScanObjectNN (OBJ) test set. The first set of results was taken from Uy et al. (2019), while the last four rows show the classification accuracy of PointNet vanilla with the different pooling operations. The results show that mode pooling (TQ, RN, and HS) has higher classification accuracy than max pooling. The results also show that when training on CAD models and testing on real-world data, mode pooling generalizes better than the other compared networks.

Table 3 shows the testing and training times, and the used GPU memory for the PointNet with different pooling operations including max, RANSAC, histogram, and Truncated Quadratic (TQ). For comparison, the feature map (shown by an orange block in Fig. 1A) dimensions for all pooling operations were set to '$10 \times 1024 \times 2048$' and '$10 \times 512 \times 512$' for the sizes of the batch, number of points, and number of features, respectively. For RANSAC, the number of hypotheses m was set to 0.2 of the total number of points. With a such number of hypotheses, and for a '$10 \times 512 \times 512$' tensor, 4Gb of GPU memory was used to train the network. The training time for one epoch was less than a minute, while its testing time was only 7 seconds. TQ requires only 0.7Gb of GPU memory and a much longer training time. For a tensor with a size of '$10 \times 1024 \times 2048$', looping was required to use RANSAC on a 12GB GPU, which affected both the training and testing times. Histogram only required 9 seconds for training one epoch and 3 seconds to finish testing, almost 100 times faster than TQ. The testing and training speeds are as fast as using max pooling. These results show that histogram pooling is significantly faster than the other robust approaches and can replace max pooling without sacrificing speed.

4.4. Classification robustness to outliers

In this section, we test the robustness of the proposed pooling operations against outliers. Visual data often contain outliers as there are imperfections in the scanning methods or processing pipelines such as the multi-view reconstruction of 3D models. An example of those outliers is the background elements in the ScanObjectNN dataset. We examined the effect of outliers on the classification accuracy of different techniques, and in particular their remaining influence after applying different pooling operations.

We considered two outliers scenarios, uniformly distributed outliers, and structured outliers (pseudo). In the first scenario, outliers were simulated by adding uniformly distributed points in the unit cube to the ModelNet40 test dataset, with ratios varying from 0 to 50% of the total number of object's points. We present the results of our experiments with added outliers in Fig. 5. The tested models are Oct-net, Pl-Net3D, and PoinNet (vanilla) with several pooling operations including histogram, RANSAC, and Truncated quadratic. We test more methods in the

Table 1
The classification accuracy on ModelNet40, ScanObjectNN, and ShapeNet datasets.

Method	Input	MN40	SC	SHPNT
PL-Net3D		86.6	70	78
VoxNet		86	80.9	80
PointNet		89.2	82	82.3
DGCNN		92.2	81	82.3
CurveNet		93.8	85	83.9
PointNet+	Points	91.8	85	83.9
PointCNN		92	88	83
PointNet* + Max		87	82	81.5
PointNet* + TQ		83.7	74	78.0
PointNet*+HS (ours)		83.7	79	77.7
PointNet*+RN (ours)		81.6	76	77.9
PointNet* + Max	Points	88.6	82	83.7
PointNet*+TQ	&	87.7	83	82.3
PointNet*+HS (ours)	normals	85.2	82	80.3
PointNet*+RN (ours)		84.8	81.2	81.2

* PointNet vanilla, + indicates the used pooling operation.

Table 2
The classification accuracy when training on ModelNet40 and testing on ScanObjectNN.

Method	OBJ
3DmFV	30.9
PointNet	42.3
SpiderCNN	44.2
PointNet+	43.6
DGCNN	49.6
PointCNN	32.2
HS (ours)	**50.2**
MAX	47.1
RN (ours)	48.6
TQ	**50.5**

Table 3
Pooling operations versus GPU usage, testing and training times for two tensor sizes.

Pooling	GPU usage	Tensor size	Testing time	Train time (one epoch)
RN	4Gb		7 s	38 s
TQ	0.7Gb	10 ×	1 m	7 m
HS	0.5Gb	512 × 512	**1 s**	**5 s**
Max	0.5Gb		**1 s**	8 s
RN	12Gb		2 m	9 m
TQ	1.5Gb	10 ×	2 m	15 m
HS	2Gb	1024 × 2048	**3 s**	**9 s**
Max	2Gb		**3 s**	**9 s**

RN: RANSAC, TQ: truncated quadratic, HS: histogram.

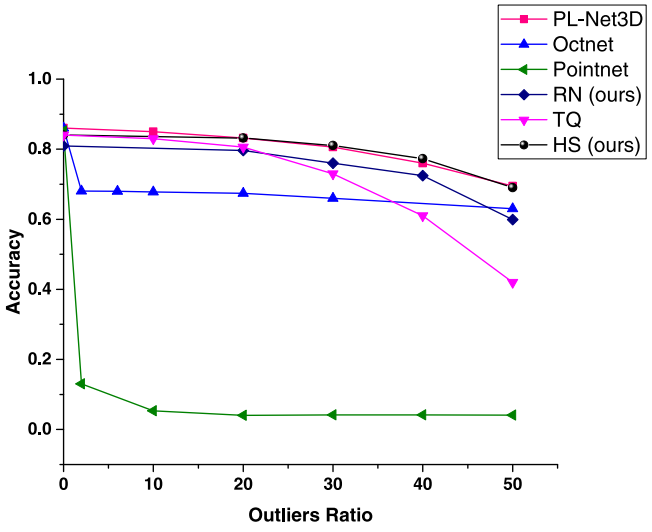


Fig. 5. Classification accuracy versus outlier ratio.

appendix.

Fig. 5 shows that the classification accuracy of PointNet with the max pooling drops significantly as outlier ratio increases. Max pooling selects outliers instead of the original object points, as shown in Fig. 1, because outliers are the furthest points from the object center (maximum radius) and that causes PointNet to miss-classify. However, the proposed pooling operations look for data clusters/dense areas that represent the object points and as such, the proposed framework is not affected by outliers. This point is highlighted in Fig. 1C.

TQ pooling scored a classification accuracy of 40% at 50% outliers ratio, while histogram pooling achieved significantly better results with 70% classification accuracy at the same outliers ratio. The performance of histogram pooling in terms of robustness to outliers is similar to Pl-Net3D, with the advantage of being much faster. The inference time for Pl-Net3D is **2.7s**, while it is **0.001s** for our method (about 2000 times faster). The classification accuracy of using Oct-Net and RANSAC pooling at 50% outliers drops to around 60%.

Other methods such as PointNet++, KPConv, and CurveNet showed similar behavior to PointNet where the classification accuracy drops to less than 10% at 50% outliers. We show the classification accuracy of PointNet++, KPConv, and CurveNet in the appendix.

In the second outliers scenario, we test the robustness of our proposed pooling operations on pseudo (structure) outliers. For this experiment, we use the ScanObjectNN dataset. The ScanObjectNN dataset contains scenes that have the point cloud of an object in addition to the point cloud of background elements or parts of nearby objects. When training, we only use the point clouds of objects. When testing, we use the point cloud of objects and the point cloud of background and parts of nearby objects as outliers (Fig. 4e). The results are shown in Table 4. The total number of outlier points reported in the dataset

Table 4
Classification accuracy on ScanObjectNN (OBJ) for objects with pseudo outliers (BG).

Method	OBJ +BG
PL-Net3D	48
PointCNN	43
CurveNet	49
DGCNN	46
PointNet+	49
HS	**67**
MAX	61
RN	62
TQ	59

reaches more than 80% of the original object points in some scenarios. We also augment all points with small jittering. The results show that the classification accuracy using the max, HS, TQ, and RN pooling operations reaches 61%, 67%, 59%, and 62% respectively, while The classification accuracy of state-of-art methods such as KPCONV, PointCNN, or Curvenet drops to around 50%. The results show that both RN and HS pooling operations achieve better performance than compared methods.

Fig. E.1 in the appendix shows the confusion matrices for PointNet with the HS and max pooling operations. Comparing those figures show that the overall classification accuracy when using all outliers mainly drops because of the low classification accuracy of two objects, chairs, and tables. The two objects have a large number of outliers ratio (ranges between 50% to 70%) which could be the reason for the miss-classification, another reason is the high similarity between some objects when outliers exist (i.e. table and desk).

4.5. Classification robustness to noise

In this section, we test the robustness of the proposed pooling operations against a noisy point cloud. We introduce Gaussian noise to the ModelNet40 test set and report the classification accuracy at different noise levels in Fig. 6. The added noise standard deviations range from 2% to 10% as seen in the figure. TQ and histogram pooling methods outperformed all the other methods. All TQ, Octnet, HS, and PL-Net3D showed similar robustness for noise levels up to 0.06 with RN robustness being slightly less. However, TQ, followed by HS and RN, showed better robustness at later noise levels. TQ scored classification accuracies of 81% and 76% at 0.06 and 0.1 noise levels respectively, while histogram pooling scored classification accuracies of 80% and 70% at 0.06 and 0.1 noise levels respectively. Pl-Net3D scored 62% classification accuracy at 0.1 noise level, followed by Oct-Net with 59%. RANSAC

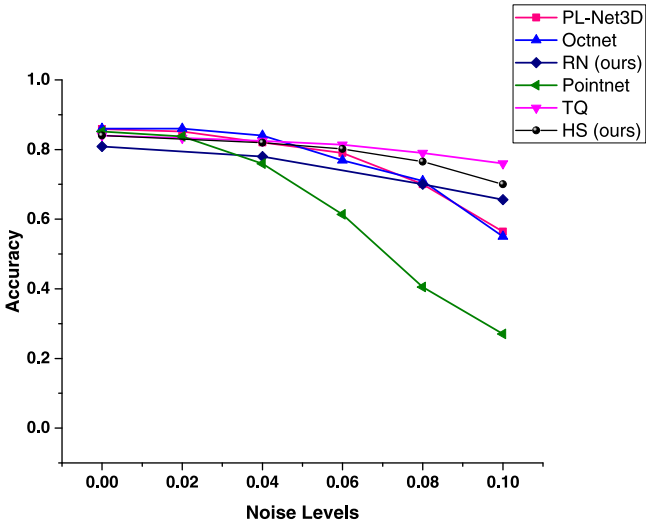


Fig. 6. Classification accuracy versus noise.

performance against noise also outperforms Pl-Net3D and Oct-Net with a classification accuracy of 66% at 0.1 noise level. Other methods such as PointNet++, KPConv, and CurveNet showed deteriorated performance against noise where the classification accuracy dropped to less than 10% at 10% noise.

4.6. Classification robustness to random point dropout

In this section, we report the robustness of the proposed pooling operations against random point dropout. We performed random point dropout to the ModelNet40 testing set with values ranging from 50% to 90%. The classification performance of different methods is shown in Fig. 7. PointNet with max pooling showed the highest robustness up to 70% random point dropout, however using the TQ pooling showed the highest robustness at the higher percentages, followed by RANSAC and histogram pooling. Both TQ and histogram methods only drop by 1 1.5% at 50% points dropout, while Pl-Net3D drops by 2.5%. OctNet performance deteriorates rapidly after 50% dropout. The classification accuracy of PointNet++, KPConv, and CurveNet was less than 40% at 90% missing points.

4.7. Segmentation robustness to outliers

In this section, we report the robustness of the proposed pooling operations against outliers for the segmentation task. We corrupted the testing part of the ShapeNet dataset with different ratios of outliers to test the performance of PointNet with different pooling operations. Table 5 shows the results of mean IoU (mIoU) and per-category scores (note that mIoU is calculated only for inliers) of PoinNet and some state-of-art methods. When there are no outliers, histogram pooling achieves 78% mIoU, max pooling scores 83%, PointCNN, DGCNN, and KPCONV score 85%, CurveNet scores 86%, and finally, RANSAC and TQ score 82%. However, when outliers are added, mIoU drops significantly for most methods. In contrast, HS, RN, and TQ mIoU almost remain constant for different outlier ratios. Comparing the results for TQ, HS, and RN show that RN only drops by 2% at 50% outliers, while TQ drops by 12% at the same outliers level. HS achieves better robustness than TQ at high outliers levels and only drops by 4% at 50% outliers. While TQ shows good robustness up to 30% outliers compared to HS. RN shows similar Robustness to TQ for outlier levels below 30% but overcomes both HS at TQ at higher outlier levels.

The above results indicate that the global features generated by histogram pooling carried robust information about the shape of the object. Unlike max pooling, histogram, RANSAC, and TQ pooling operations enabled the decoder part of the network to segment the objects correctly. Fig. 8 shows instances of segmented objects with outliers. As can be seen, segments with histogram pooling are almost similar to the original object segments, while with max pooling, many segments are misclassified.

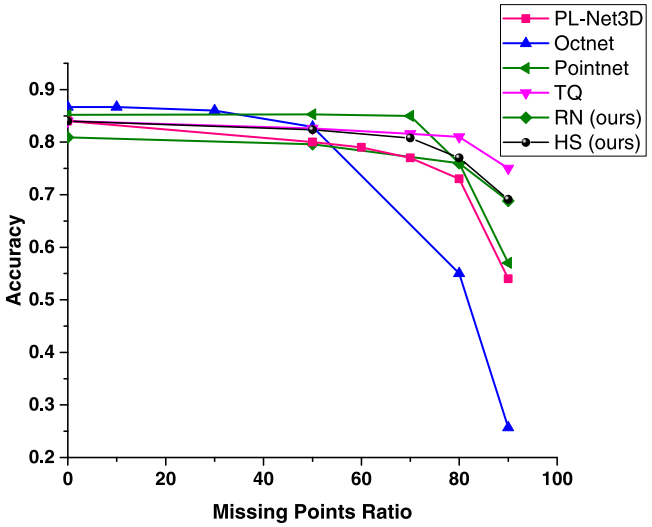


Fig. 7. Classification accuracy versus missing points.

Table 6 shows the testing time for the different pooling operations. Comparing both Tables 5 and 6 show that both HS and RN are extremely faster than TQ and are able to achieve higher robustness at high outlier levels while having similar robustness to TQ at lower outliers levels.

4.8. Normals estimation

In this section, we report the robustness of the proposed pooling operations against noise for the points normal estimation task. We trained the segmentation networks of some state-of-art methods to predict the point's normal vector (the last layer was modified to predict the normal vector for each point). We used the absolute value of cosine distance as the loss, and we used the ModelNet40 dataset to evaluate the methods. We perturb the point cloud with different noise levels, and we report the average cosine-distance error in Table 7.

Noise has a big influence on any normal estimation process, thus it is essential to validate the robustness of any method on this type of data perturbation. Table 7 shows the robustness of DGCNN, CurveNet, and PoinNet with Max, TQ, RN, and HS pooling operations. PoinNet, in general, shows better robustness than other compared method, and the use of TQ, RN, and HS help achieve better results. HS shows better robustness at high noise levels, while RN, followed by TQ, shows better robustness at low noise levels.

5. Sensitivity analysis

In this section, we evaluate the performance of the proposed pooling layer as part of the PointNet and DGCNN methods. The study also includes the sensitivity analysis of the histogram bin size on the classification accuracy. In these experiments, the data is corrupted by 10% additive noise and 50% outliers.

The performance of the histogram pooling layer within different network structures is shown in Table 8. The first row shows the classification accuracy of the PointNet when it uses the two transformation networks that are designed to estimate rotation and translation - referred to as PointNet(1). The last two rows show the classification accuracy of PointNet vanilla (PointNet without transformation networks) when the histogram pooling layer is used - referred to PointNet (2). It is somewhat surprising to note that the classification accuracy of PointNet decreases by using transformation networks. The transformation networks appear to be overly sensitive to data perturbation and corruption. The last row shows that using points normal, in PointNet (2) achieves higher classification accuracy for clean data, while its robustness against outliers and noise is less than the case when it uses point coordinates.

The second row shows the classification accuracy of the original DGCNN architecture, while the third row shows the classification accuracy when the histogram pooling layer is used - referred to as DGCNN (1). The method showed higher robustness to outliers compared to the original DGCNN. In the fourth row, we modified the convolution layers of the DGCNN(1) to include only neighboring points within a certain radius (0.25 for the first convolution layer, and 2 for the rest of the convolution layers). This is called DGCNN(2) and its robustness to data perturbation and corruption has enhanced, especially for noise. In the fifth row, we modified DGCNN to include only two convolution layers, which is called DGCNN(3). The results show that using only two convolution layers achieves the highest robustness to outlier corruption and noise perturbation.

Fig. 9 shows the classification accuracy of the histogram pooling under several inlier thresholds (the threshold shown in the figure is half of the bin size). As can be seen from the figure, setting the threshold

Table 5
Segmentation results on ShapeNet part dataset. We compare PointNet vanilla with max, RANSAC, TQ and histogram pooling. The results show the importance of using robust pooling over max.

outl %	Method	mean	Airo	Bag	Cap	Car	Chair	Ear phone	Guitar	Knife	Lamp	Laptop	Motor bike	Mug	Pistol	Rocket	Skate board	Table*
	PointCNN	0.85	0.83	0.83	0.86	0.81	0.90	0.75	0.91	0.88	0.84	0.96	0.74	0.95	0.83	0.62	0.79	0.82
0	DGCNN	0.85	0.83	0.85	0.76	0.90	0.91	0.75	0.91	0.87	0.82	0.96	0.64	0.95	0.81	0.59	0.75	0.82
	KPCONV	0.85	0.83	0.85	0.85	0.80	0.90	0.77	0.91	0.88	0.79	0.96	0.75	0.96	0.86	0.62	0.80	0.83
	CurveNet	**0.86**	0.84	0.82	0.90	0.80	0.91	0.79	0.91	0.88	0.84	0.96	0.63	0.95	0.80	0.57	0.76	0.83
	max	0.83	0.83	0.72	0.80	0.74	0.89	0.68	0.91	0.84	0.80	0.95	0.64	0.90	0.82	0.53	0.71	0.81
	HS	0.78	0.77	0.62	0.70	0.62	0.84	0.67	0.88	0.80	0.75	0.93	0.41	0.84	0.70	0.47	0.61	0.77
	RN	0.82	0.79	0.75	0.74	0.68	0.87	0.71	0.90	0.84	0.79	0.94	0.56	0.89	0.78	0.50	0.69	0.80
	TQ	0.82	0.81	0.73	0.72	0.71	0.88	0.67	0.90	0.82	0.78	0.95	0.60	0.92	0.79	0.49	0.70	0.81
	PointCNN	0.12	0.10	.04	.04	0.05	0.21	0	0.03	0.20	0.05	0.09	0.05	0.01	0	0	0.03	0.15
5	DGCNN	0.53	0.50	0.55	0.68	0.40	0.56	0.32	0.47	0.41	0.51	0.89	0.23	0.86	0.42	0.24	0.30	0.55
	KPCONV	0.76	0.79	0.80	0.32	0.76	0.88	0.72	0.72	0.71	0.77	0.27	0.75	0.95	0.71	0.45	0.65	0.72
	CurveNet	0.25	0.11	0.00	0.00	0.07	0.42	0.13	0.00	0.00	0.47	0.00	0.08	0.10	0.01	0.00	0.16	0.26
	max	0.37	0.12	0.45	0.27	0.10	0.64	0.34	0.13	0.60	0.18	0.43	0.25	0.74	0.07	0.15	0.15	0.40
	HS	0.78	0.76	0.65	0.72	0.62	0.84	0.69	0.88	0.79	0.75	0.93	0.41	0.83	0.71	0.46	0.60	0.77
	RN	0.81	0.79	0.71	0.76	0.68	0.87	0.71	0.89	0.83	0.79	0.95	0.56	0.90	0.79	0.53	0.71	0.80
	TQ	**0.82**	0.80	0.72	0.75	0.70	0.88	0.67	0.90	0.83	0.78	0.95	0.57	0.92	0.81	0.50	0.69	0.80
	PointCNN	0.09	0.07	.04	.04	0.06	0.11	0	0.11	0.30	0.05	0	0.05	0.01	0	0	0.06	0.13
20	DGCNN	0.44	0.40	0.52	0.65	0.33	0.44	0.32	0.45	0.44	0.51	0.75	0.17	0.75	0.37	0.26	0.28	0.42
	KPCONV	0.50	0.66	0.62	0.18	0.53	0.73	0.54	0.40	0.45	0.73	0.15	0.60	0.85	0.38	0.27	0.48	0.20
	CurveNet	0.18	0.1	0.00	0.00	0.06	0.24	0.12	0.00	0.00	0.49	0.00	0.06	0.02	0.01	0.00	0.16	0.17
	max	0.30	0.10	0.45	0.17	0.14	0.43	0.38	0.05	0.64	0.08	0.40	0.17	0.54	0.09	0.17	0.14	0.37
	HS	0.78	0.77	0.64	0.71	0.61	0.84	0.68	0.88	0.78	0.75	0.93	0.41	0.84	0.72	0.46	0.60	0.77
	RN	**0.81**	0.78	0.71	0.74	0.69	0.87	0.65	0.89	0.83	0.77	0.95	0.55	0.91	0.80	0.49	0.68	0.80
	TQ	**0.81**	0.79	0.75	0.81	0.70	0.88	0.76	0.89	0.81	0.80	0.94	0.56	0.91	0.80	0.49	0.67	0.80
	HS	0.77	0.74	0.62	0.69	0.59	0.84	0.67	0.87	0.77	0.75	0.92	0.40	0.83	0.72	0.46	0.58	0.77
30	RN	**0.81**	0.79	0.71	0.75	0.67	0.87	0.67	0.88	0.83	0.79	0.94	0.54	0.89	0.79	0.54	0.70	0.80
	TQ	0.79	0.76	0.73	0.78	0.66	0.86	0.60	0.89	0.83	0.74	0.93	0.49	0.91	0.80	0.44	0.62	0.78
	HS	0.76	0.72	0.57	0.70	0.56	0.84	0.66	0.87	0.75	0.74	0.92	0.37	0.83	0.71	0.50	0.57	0.77
40	RN	**0.80**	0.786	0.70	0.74	0.66	0.87	0.73	0.87	0.83	0.79	0.95	0.50	0.88	0.78	0.53	0.69	0.80
	TQ	0.76	0.70	0.72	0.73	0.62	0.84	0.56	0.88	0.83	0.71	0.92	0.47	0.90	0.78	0.42	0.55	0.73
	HS	0.74	0.67	0.57	0.66	0.53	0.83	0.68	0.86	0.70	0.74	0.91	0.38	0.82	0.68	0.49	0.55	0.75
50	RN	**0.79**	0.76	0.70	0.74	0.64	0.86	0.66	0.84	0.82	0.77	0.93	0.49	0.87	0.77	0.56	0.65	0.77
	TQ	0.70	0.62	0.72	0.73	0.57	0.80	0.52	0.87	0.81	0.68	0.91	0.47	0.89	0.73	0.42	0.50	0.65

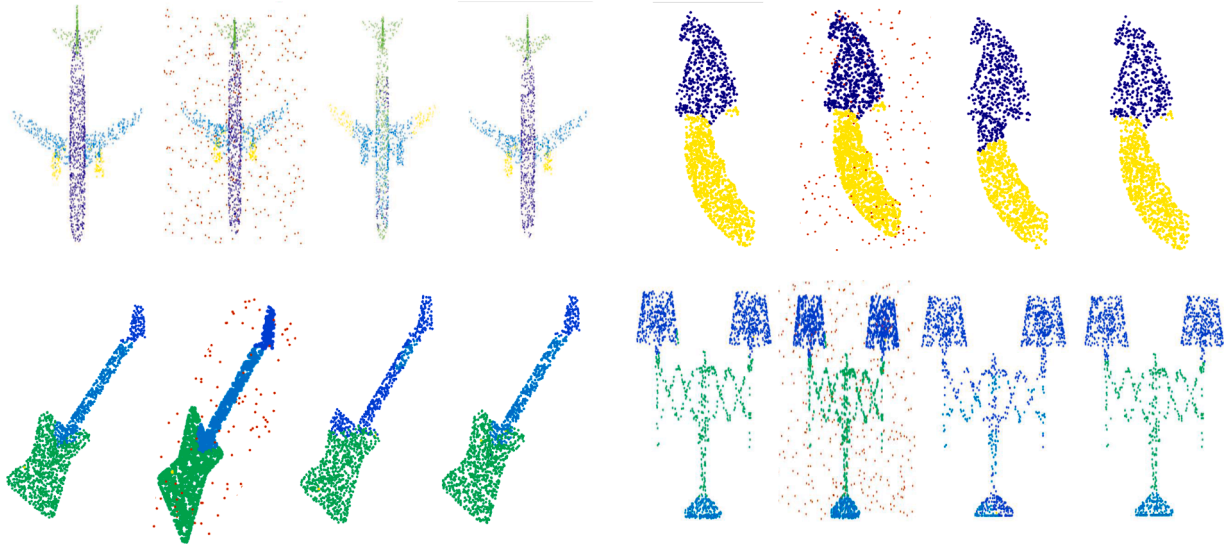


Fig. 8. Four samples of ShapeNet part dataset showing the original objects, the same objects corrupted by outliers, PoinNet with max pooling segmentation results, and finally PoinNet with histogram pooling segmentation results (ours).

Table 6
Pooling operations versus GPU usage and testing time for part segmentation.

Pooling	GPU usage	Tensor size	Testing time (s)
			TQ
	3.1 Gb	$16 \times 2048 \times 2048$	2.34
RN	8.9 Gb		0.039
MAX	1.9 Gb		0.003
HS	2.4 Gb		0.022

Table 7
Normal estimation error at different noise levels. Error is calculated based on the average cosine distance.

Method	Noise levels						
	0.002	0.02	0.04	0.06	0.08	0.1	0.2
CurveNet	0.492	0.735	0.827	0.863	0.883	0.897	0.926
DGCNN	0.671	0.718	0.799	0.83	0.85	0.862	0.9
Max	**0.313**	0.479	0.607	0.687	0.736	0.764	0.833
TQ	0.36	0.461	0.559	0.628	0.678	0.718	0.818
RN	0.36	**0.458**	**0.557**	0.629	0.685	0.724	0.808
HS	0.459	0.504	0.567	**0.62**	**0.663**	**0.697**	**0.794**

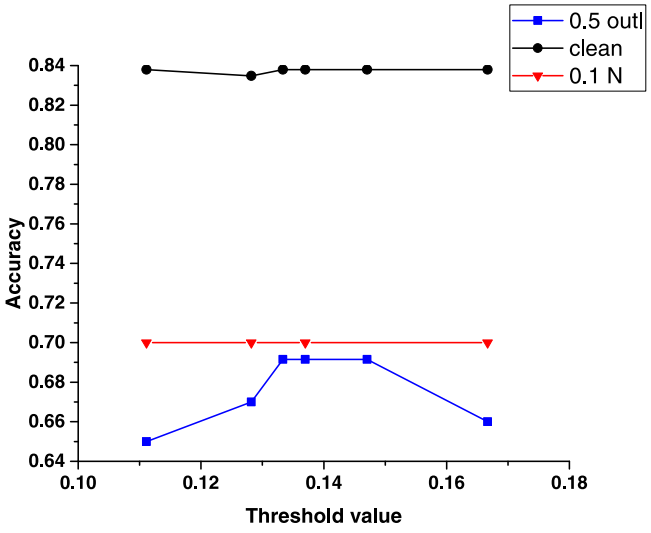


Fig. 9. Classification accuracy versus histogram threshold/bin size.

Table 8
classification performance on ModelNet40 with **histogram** pooling, clean: clean objects, outl: objects with 50% outliers, noise: objects with 10% noise

Method	Input	clean	outl	noise
PointNet(1)		69	58	63
DGCNN	p	**92**	5	5
DGCNN(1)		84	39	3
DGCNN(2)		85	51	39
DGCNN(3)		85	**70**	63
PointNet(2)		84	69	**70**
	P+n	85	60	13

between 0.13-0.15 provides the highest robustness to outliers, while the classification accuracy of clean and noisy data remains constant. RN threshold values are shown in Table F.1 in the appendix.

6. Limitation and future work

The above results show that the robustness of PoinNet was

Table D.1
The classification accuracy on the clean ModelNet40 (MN40), the ModelNet40 perturbed with 0.1 Gaussian noise, the ModelNet40 corrupted with 50% outliers, and 90% missing points, respectively.

Method	Input	MN40	OUT	Noise	Dropout
PL-Net3D		86.6	70	60	50
PointNet		89	4	27	57
DGCNN		**92.2**	5	5	18
PointNet+	Points	91.8	2	2	30
PointCNN		91	20	4	7
KPConv		90	4	4	12
CurveNet		93.8	4	5	26
RSCNN		80.5	**72**	63	58
Welsch		82.4	37	69	68
Huber		81.8	4	71	71
TQ		83.7	51	**73**	**75**
HS (ours)		83.7	69	70	69
RN (ours)		81.6	60	66	69

Table F.1
The Classification accuracy for different RN threshold values. We report the classification accuracy on the clean/original ModelNet40 (MN40), the ModelNet40 perturbed with 0.1 Gaussian noise (noise), and the ModelNet40 corrupted with 50% outliers (outl).

Threshold	MN40	outl	noise
0.05	80.5	64	63
0.11	82.3	60	64
0.13	80.9	59	64
0.143	81.6	60	66
0.148	81.2	58	62
0.168	82.4	61	64
0.2	82	61	68
0.25	82	61	65
0.6	81.5	47	71

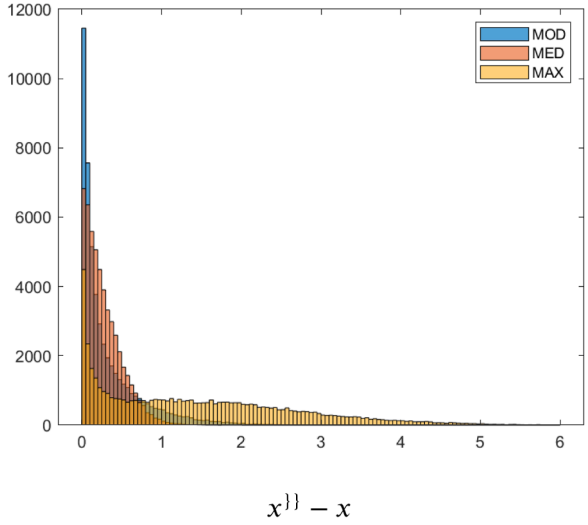


Fig. C.1. Difference in pooling output between features of the clean (x) and its outlier corrupted point clouds ($x^{\cdot\cdot}$) of an object for mode, max, and median pooling operations.

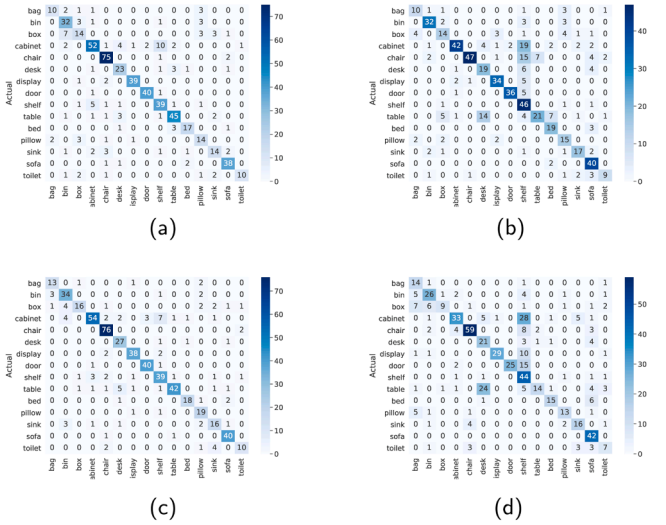


Fig. E.1. The confusion matrix of the ScanObjectNN dataset for PoinNet with (a) histogram pooling for clean objects, (b) histogram pooling for objects with background points, (c) max pooling for clean objects, and (d) max pooling for objects with background points.

Table F.2
The Classification accuracy for different TQ threshold values. We report the classification accuracy on the clean/original ModelNet40 (MN40), the ModelNet40 perturbed with 0.1 Gaussian noise (noise), and the ModelNet40 corrupted with 50% outliers (outl). We also show the time, in minutes, required to train one epoch.

Threshold	MN40	outl	noise	Time
0.2	5	04	5	120
0.5	5	04	5	120
1	83.7	51	73	30
1.5	83	27	77	20
2	81.5	9	74	13

significantly improved by using RANSAC or histogram based pooling layers. PointNet with any of these pooling layers can tolerate a large number of outliers and noise levels compared to the network with the max pooling layer. The first shortcoming of the proposed robust pooling operations is the requirement of setting their thresholds. However, this paper already provides the threshold values for different tasks such as classification and segmentation. Another shortcoming of using RANSAC is that the memory requirement grows rapidly with data size, which could limit its usage in applications with large point cloud datasets. In addition to the above shortcomings, the proposed pooling layers are only suitable for PointNet-based architectures or architectures with global pooling operations. Despite those shortcomings, both methods showed promising results and can open a window for future improvement in this area.

Future work includes using the proposed pooling operations in other tasks such as point cloud registration. Also, future work might address the above shortcomings by modifying the proposed pooling operations to be included in any network architecture, not only in PointNet-based architectures.

7. Conclusion

We presented two pooling operations that are robust to data corruption. The proposed pooling layers use histogram and RANSAC algorithm to look for clusters in data as clusters are indicatives of models. We tested those pooling layers with frameworks such as Point-based and graph-based neural networks that have a global pooling layer such as PointNet and DGCNN. For the task of classification, our results showed that the robustness of the proposed frameworks is significantly higher compared to max pooling. When comparing our proposed pooling layers with robust state-of-the-art methods such as M-estimators, our histogram pooling was much faster and significantly more robust to outliers, with comparable robustness to noise and random point dropout. Compared to PL-Net3D, our histogram pooling was also significantly faster and more robust to noise and random point dropout, while we achieve similar robustness to outliers. For the tasks of part segmentation and normals estimation, both RN and HS showed comparable results to TQ with better performance in some cases, with the advantage of being much faster.

CRediT authorship contribution statement

Ayman Mukhaimar: Investigation, Methodology, Writing – review & editing, Conceptualization. **Ruwan Tennakoon:** Supervision, Writing – review & editing, Conceptualization. **Reza Hoseinnezhad:** Supervision, Writing – review & editing, Conceptualization. **Chow Yin Lai:** Supervision, Writing – review & editing, Conceptualization. **Alireza Bab-Hadiashar:** Supervision, Writing – review & editing, Project administration, Conceptualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Appendix A. Density approximate and marginalization

Consider a D-dimensional feature space $X \subseteq R^D$ where D is the number of features. Denoting the joint density of the features by $p(x)$, let's assume it is approximated by a Gaussian mixture:

$$p(x) = \sum_{m=1}^{M} w_m N\left(x; \mu^{(m)}, \sum^{(m)}\right) \tag{A.1}$$

where w_m's are normalized importance weights, i.e. $\sum_{m=1}^{M} w_m = 1$, and $\mu^{(m)}$ and $\sum^{(m)}$ are the mean vector (Dx1) and covariance matrix (DxD) for the m_{th} Gaussian component whose density is given by:

$$N\left(x; \mu^{(m)}, \sum^{(m)}\right) = \frac{\exp\left(-0.5\left(x-\mu^{(m)}\right)^T \sum^{(m)} \left(x-\mu^{(m)}\right)\right)}{\left[\sqrt{2\pi} * det\left(\sum^{(m)}\right)\right]^D} \tag{A.2}$$

Let's denote the elements of the mean and covariance of the m_{th} Gaussian component as :

$$\mu^{(m)} = \begin{bmatrix} \mu_1^{(m)} \\ \mu_2^{(m)} \\ \dots \\ \mu_D^{(m)} \end{bmatrix}, \sum^{(m)} = \begin{bmatrix} \sigma_1^{2(m)} & \sigma_{12}^{(m)} & \dots & \sigma_{1D}^{(m)} \\ \sigma_{21}^{(m)} & \sigma_2^{2(m)} & \dots & \sigma_{2D}^{(m)} \\ \dots & \dots & \dots & \dots \\ \sigma_{D1}^{(m)} & \sigma_{D2}^{(m)} & \dots & \sigma_D^{2(m)} \end{bmatrix} \tag{A.3}$$

It has been proven that the marginal distribution over a subset of the features is also Gaussian with its mean and covariance being the original from which only the rows (and the columns for $\sum$) corresponding to the feature subset are retained and the rest are removed.[1]

Hence, for any i= 1,..., D, the uni-variate marginal density of the math component is $N(x_i; \mu_i^{(m)}, \sigma_i^{(m)})$ and

$$p(x_i) \approx \sum_{m=1}^{M} w_m N\left(x_i; \mu_i^{(m)}, \sigma_i^{2(m)}\right) \tag{A.4}$$

Appendix B. Maxima of joint and marginal densities

Lets assume that the first component $m = 1$ of the Gaussian mixture in Eq. (A.1) is dominant; i.e. $\forall m > 1 \; ; \; w_m >> w_1$.

We assume that such a dominance leads to the location of the peak of the first component being the location of the overall peak, i.e., the MAP estimate:

$$\widehat{x}_{\text{MAP}} = \mu^{(1)} = \begin{bmatrix} \mu_1^1 \\ \dots \\ \mu_D^1 \end{bmatrix} \tag{B.1}$$

with the above assumption (dominance of the first Gaussian component), from Eq. (A.4), the peak of the point of the marginal density will be at the mean of the first Gaussian component too, i.e. at $\mu_i^{(1)}$. Thus, the MMAP estimate, constructed by stacking the $\mu_i^{(1)}$ values will equal the MAP estimate.

Appendix C. Robustness of mode pooling

To compare the robustness of the max, mean, median, and mode in pooling operations, we first randomly selected 50 (out of 2048) feature vectors (of size 1024) of the feature map (shown in Fig. 1A by the orange box). The experiment was repeated for both a clean and an object corrupted with 50% outliers. The above-mentioned pooling operations were applied to both feature collections. Pooling outputs of the clean object were subtracted from the outputs of the corrupted object and the differences were plotted as shown in Fig. C.1. Average and median pooling were very similar and the median is only plotted. The figure shows that mode pooling has the lowest output difference between clean and corrupted data, indicating significant robustness to the presence of outliers.

[1] https://math.stackexchange.com/questions/3832119/prove-that-the-distribution-of-marginal-vectors-are-also-multivariate-normal/3832137#3832137

Appendix D. The classification accuracy on ModelNet40 with the presence of data perturbation and corruption

We compare the robustness of several state-of-art methods to data corruptions in Table D.1. The table shows that several methods such as DGCNN, PointNet++, PoinctCNN, and CurveNet have low robustness to data corruption. The table also shows that Welsch and Huber Gould et al. (2019) have low robustness to outliers, while RSCNN Mukhaimar et al. (2022) and PL-Net3D show similar robustness to TQ, RN, and HS. Overall, HS shows the best performance in terms of robustness to outliers, noise, and missing points.

Appendix E. The confusion matrix of the ScanObjectNN dataset

We show the confusion matrix of the ScanObjectNN dataset for PoinNet with histogram and max pooling in Fig. E.1. Comparing both figures for objects with background data indicates that both methods misclassify tables to be desks due to the high similarity between both objects when background data exists.

Appendix F. Classification accuracy versus pooling threshold

Tables F.1 and F.2 show the classification accuracy for the RANSAC and TQ pooling operations for several thresholds, respectively. The classification accuracy was reported for the ModelNet40 dataset. As shown in Table F.1, the classification accuracy does not vary much by changing the RN threshold between 0.05-0.25. However, setting the threshold to higher values, such as 0.6, increases the robustness to noise, while reducing the robustness to outliers. Similar behavior is also observed when RANSAC is used for geometric fitting, where increasing the threshold values means that RANSAC can tolerate more noise levels and at the same time increases the number of outliers selected (wrongly). Table F.2 shows the classification accuracy and the time required to train one epoch for different TQ threshold values. Reducing the threshold values increases the training time. This is in contrast to RN training time, which is not dependant on its threshold value (the training time for one epoch using RN is around 15 minutes for the different threshold values). Additionally, the results show that using lower TQ threshold values causes the optimizer to stuck in local minima, and thus no solution was found (the classification accuracy is almost zero).

References

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., & Yu, F. (2015). ShapeNet: An information-rich 3D model repository. *Technical Report*. Stanford University — Princeton University — Toyota Technological Institute at Chicago. arXiv preprint arXiv:1512.03012 [cs.GR].

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al., 2015b. ShapeNet: An information-rich 3D model repository. arXiv preprint arXiv:1512.03012.

Chen, C., Li, G., Xu, R., Chen, T., Wang, M., & Lin, L. (2019). ClusterNet: deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. *The IEEE conference on computer vision and pattern recognition (CVPR)*.

Esteves, C., Allen-Blanchette, C., Makadia, A., & Daniilidis, K. (2018). Learning SO(3) equivariant representations with spherical CNNs. *Proceedings of the European conference on computer vision (ECCV)* (pp. 52–68).

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM, 24*(6), 381–395.

Gould, S., Hartley, R., Campbell, D., 2019. Deep declarative networks: A new hope. arXiv preprint arXiv:1909.04866.

Klokov, R., & Lempitsky, V. (2017). Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models. *Proceedings of the IEEE international conference on computer vision* (pp. 863–872).

Leroy, A. M., Rousseeuw, P. J., 1987. Robust regression and outlier detection (RROD).

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., & Chen, B. (2018). PointCNN: Convolution on X-transformed points. *Advances in Neural Information Processing Systems, 31*, 820–830.

Maturana, D., & Scherer, S. (2015). VoxNet: A 3D convolutional neural network for real-time object recognition. *2015 IEEE/RSJ International conference on intelligent robots and systems (IROS)* (pp. 922–928). IEEE.

Mukhaimar, A., Tennakoon, R., Lai, C. Y., Hoseinnezhad, R., & Bab-Hadiashar, A. (2019). Comparative analysis of 3D shape recognition in the presence of data inaccuracies. *2019 IEEE International conference on image processing (ICIP)* (pp. 2471–2475). IEEE.

Mukhaimar, A., Tennakoon, R., Lai, C. Y., Hoseinnezhad, R., & Bab-Hadiashar, A. (2019). PL-Net3D: Robust 3D object class recognition using geometric models. *IEEE Access, 7*, 163757–163766.

Mukhaimar, A., Tennakoon, R., Lai, C. Y., Hoseinnezhad, R., & Bab-Hadiashar, A. (2022). Robust object classification approach using spherical harmonics. *IEEE Access, 10*, 21541–21553.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). PointNet: Deep learning on point sets for 3D classification and segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 652–660).

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems* (pp. 5099–5108).

Ramasinghe, S., Khan, S., Barnes, N., & Gould, S. (2019). Representation learning on unit ball with 3D roto-translational equivariance. *International Journal of Computer Vision*, 1–23.

Riegler, G., Osman Ulusoy, A., & Geiger, A. (2017). OctNet: Learning deep 3D representations at high resolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3577–3586).

Shi, B., Bai, S., Zhou, Z., & Bai, X. (2015). DeepPano: Deep panoramic representation for 3-D shape recognition. *IEEE Signal Processing Letters, 22*(12), 2339–2343.

Siddiqi, K., Zhang, J., Macrini, D., Shokoufandeh, A., Bouix, S., & Dickinson, S. (2008). Retrieving articulated 3-D models using medial surfaces. *Machine Vision and Applications, 19*(4), 261–275.

Su, H., Maji, S., Kalogerakis, E., & Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3D shape recognition. *Proceedings of the IEEE international conference on computer vision* (pp. 945–953).

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

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2019). Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics (TOG), 38*(5), 1–12.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1912–1920).

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1912–1920).

Xiang, B., Tu, J., Yao, J., & Li, L. (2019). A novel octree-based 3-D fully convolutional neural network for point cloud classification in road environment. *IEEE Transactions on Geoscience and Remote Sensing*.

Xiang, T., Zhang, C., Song, Y., Yu, J., & Cai, W. (2021). Walk in the cloud: Learning curves for point clouds shape analysis. *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 915–924).

Yi, L., Kim, V. G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., & Guibas, L. (2016). A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics (ToG), 35*(6), 1–12.

Zhang, K., Hao, M., Wang, J., de Silva, C. W., Fu, C., 2019a. Linked dynamic graph CNN: Learning on point cloud via linking hierarchical features. arXiv preprint arXiv:1904.10014.

Zhang, Z., Hua, B.-S., Rosen, D. W., Yeung, S.-K., 2019b. Rotation invariant convolutions for 3D point clouds deep learning. arXiv preprint arXiv:1908.06297.

Zhou, Y., & Tuzel, O. (2018). VoxelNet: End-to-end learning for point cloud based 3D object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4490–4499).