Scalability of Learning Tasks on 3D CAE Models using Point Cloud Autoencoders

Thiago Rios*, Patricia Wollstadt*, Bas van Stein†, Thomas Bäck†,
Zhao Xu‡, Bernhard Sendhoff* and Stefan Menzel*
* Honda Research Institute Europe GmbH, Carl-Legien-Str. 30, 63073 Offenbach, Germany
†Leiden Institute of Advanced Computer Science (LIACS), Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
‡NEC Laboratories Europe, Kurfürsten-Anlage 36, 69115 Heidelberg, Germany
Email: {thiago.rios, patricia.wollstadt, bernhard.sendhoff, stefan.menzel}@honda-ri.de@honda-ri.de,
{b.van.stein, t.h.w.baeck}@liacs.leidenuniv.nl, zhao.xu@neclab.eu

***Abstract*—Geometric Deep Learning (GDL) methods have recently gained interest as powerful, high-dimensional models for approaching various geometry processing tasks. However, training deep neural network models on geometric input requires considerable computational effort, even more so if one considers typical problem sizes found in application domains such as engineering tasks, where geometric data are often orders of magnitude larger than the inputs currently considered in GDL literature. Hence, an assessment of the scalability of the training task is necessary, where model and data set parameters can be mapped to the computational demand during training. The present paper therefore studies the effects of data set size and the number of free model parameters on the computational effort of training a Point Cloud Autoencoder (PC-AE). We further review pre-processing techniques to obtain efficient representations of high-dimensional inputs to the PC-AE and investigate the effects of these techniques on the information abstracted by the trained model. We perform these experiments on synthetic geometric data inspired by engineering applications using computing hardware with particularly recent graphics processing units (GPUs) with high memory specifications. The present study thus provides a comprehensive evaluation of how to scale geometric deep learning architectures to high-dimensional inputs to allow for an application of state-of-the-art deep learning methods in real-world tasks.**

***Keywords*—deep learning; dimensionality reduction; computer aided engineering; geometry; sampling methods**

I. Introduction

Geometric deep learning (GDL) methods have recently gained interest as powerful, high-dimensional models for the application in geometry processing tasks such as segmentation [1], classification [2], or object recognition [3], and others [4], [5]. The increasing availability of 3D data as well as powerful computing hardware drive the adaption of successful deep learning architectures to the 3D- and non-Euclidean domain in general [6]. However, training deep neural network models on geometric input requires considerable computational effort and often real-world input data sizes are prohibitively large. For example, in application domains such as engineering, where 3D data is ubiquitous, typical problem sizes are orders of magnitude larger than the inputs currently considered in GDL literature. For these domains it is necessary to assess whether deep learning models can be applied in their current form, and where further development is necessary to harness the power of recent architectures for solving real-world tasks.

Central to an efficient processing of 3D data is the chosen data representation. Here, various approaches have been proposed [4], [5]: For example, volumetric approaches represent shapes as occupied voxels in a 3D grid, which due to its Euclidean structure allows for an adaption of deep learning concepts from image processing to 3D input. However, computational and storage demand for voxel representations is cubic in the input size, which severely limits the maximum achievable resolution [5], [7]. More efficient representations have been proposed in the non-Euclidean domain, such as polygon meshes and point clouds.

Polygon meshes represent 3D shapes as vertices and their Cartesian coordinates, with an embeddeded description of the connectivity between these vertices. Meshes are popular due to their efficiency in describing surfaces with high-resolution, however, translating successful concepts like shift-invariance from the 2D to the non-Euclidean domain is not straightforward [6]. Attempts to adapt these concepts have been made by replacing convolutional filters with local patch operators in the 3D domain [8], [9]. Yet, calculating local operators per node is computationally expensive and becomes infeasible for large mesh sizes. On the other hand, mesh-based approaches that do not define local operators on the shape such as spectral approaches are currently limited to topologically similar shapes, due to requirements like constant connectivity or close correspondence between vertices (e.g., [10]).

Point clouds have recently been introduced as powerful and efficient representations of shapes for geometric deep learning [11], [12]. In comparison to voxel or mesh representations, they are memory efficient while being able to preserve a high amount of geometric detail. Architectures proposed for the processing of point clouds do not require topological similarity in the input such as mesh-based models. Point clouds have further been popularized through advances in data-acquisition and 3D scanning technologies, especially in the fields of computer vision, robotics and autonomous driving, which use point clouds for the representation of objects and scenes [13]–[15]. Generating point clouds is computationally cheap, especially compared to other data generation algorithms such as meshing: they may either be acquired directly from physical objects via various data-acquisition techniques that typically require only minor post-processing for denoising and interpolation of occluded regions [16]–[18]; on the other hand, point clouds may be acquired from other 3D representations through virtual sampling of surfaces or by re-using mesh vertices, where the latter approach allows to preserve valuable information like surface normals.

The discussed characteristics make point clouds a promising representation, for example, for 3D-data from computer aided engineering (CAE), where shapes are typically represented as surface meshes. For CAE data, point clouds can be sampled virtually from an initial representation based on meshes or from non-parametric functions (e.g. NURBS) [19]. The quality of the obtained point cloud and how well it can be used in machine learning tasks then depends on the characteristics of the sampling algorithm, in particular, the density and regularity of the sampling, which determines how and which geometric information is preserved. Therefore, algorithm choice is essential for the performance of the model subsequently trained on the data.

Yet, even though point clouds are a promising data representation in geometric processing they also pose challenges for GDL. In particular, point clouds can be considered unordered sets, which requires operations on point clouds to be invariant to permutations in the ordering of input points. Several architectures adressing this problem using permutation-invariant operations wihtin the network have been proposed [11], [20], [21]. A second challenge is the high dimensionality of the input, especially if a high resolution in the input representations is required. In CAE applications this may generally be the case, even more so if the machine learning task to be solved is to learn a predictive or regression model from data [22].

Currently, the scalability of point cloud GDL architectures to input sizes allowing for the application in data analysis tasks typically encountered in various application domains is lacking. The present paper therefore investigates how point cloud models scale for increasing input size, where we extend an existing point cloud architecture and loss function to investigate a (re-)sampling scheme as a preprocessing technique in order to encode relevant geometric information more efficiently.The architecture considered is a point cloud autoencoder (PC-AE) presented in [21]. Autoencoders learn low-dimensional, latent representations of complex input data, either in order to identify latent variables underlying the data generation process or to use the low-dimensional representations as input to further machine learning tasks [23]. We perform experiments on synthetic geometric data inspired by engineering applications, where we have full control over the data generation process. We evaluate whether the PC-AE is able to recover the latent features underlying the data generation process as a function of the sampling scheme used. In particular, by focusing on the learning of constantly varying features within an object class we evaluate to what extend the learned variables are able to represent such finer geometric features, which may be relevant in tasks such as predictive modeling. We complement our experiments with an evalution of practical running times on different computing hardware, in particular recent graphics processing units (GPUs) with high memory specifications. The present study thus provides a first evaluation of how targeted sampling of input data can be used to scale geometric deep learning architectures to high-dimensional inputs to allow for an application of state-of-the-art deep learning methods in real-world tasks.

II. Literature Review

In order to address the scalability of a PC-AE to high-dimensional point clouds, it is important to first identifiy what are the constraints for training such models. Mandikal and Babu explore in [24] a deep learning architecture for handling the reconstruction of dense 3D point clouds from RGB images. The authors point out that scaling the size of the point clouds lead to a corresponding increase in the number of parameters to be trained, therefore also the difficulty to abstract the dataset. Furthermore, permutation-invariant loss functions such as the Chamfer Distance (CD) [11], [21], [25], are computed point-wise and thus might lead to prohibitive computational costs when applied to large point clouds. Starting with the first, in the following we will therefore review existing GDL architectures for processing point clouds with a focus on the number of model parameters and how this number scales as a function of input size.

Qi et al. were the first to propose an architecture taking point clouds as inputs, called PointNet [11], which consists of multiple fully connected layers with a pre-processing step to make the input representation order invariant. The work was later extended in [20] to PointNet++, which consists of stacked PointNet architectures and a clustering operation to support the identification of local features. For such networks with only *fully connected* layers and bias, we calculate the number of free parameters, N_p, as

$$N_p = H_1(N_i + 1) + \sum_{l=1}^{L-1} H_l(H_{l-1} + 1) + N_i(H_L + 1), \quad (1)$$

where L is the number of layers, H_l is the number of neurons in the layer l and N_i is the size of the input. Hence, the number of model parameters grows linearly in the input size such that training effort grows linearly in the size of point clouds and depends on the number of neurons in the layers adjacent on the input and output only. In practice, due to the spatial nature of the input, the number of parameters has to be multiplied by a factor of three because each Euclidean dimension is being treated by an individual set of fully connected layers.

In order to address unsupervised learning tasks, while significantly reducing network size, Yang et al. proposed an autoencoder architecture named FoldingNet [26]. On the encoder side, FoldingNet uses the PointNet architecture, while the decoder architecture is fundamentally different from previous approaches and uses two 3-layer perceptrons to perform an operation analogous to folding a 2D grid into the format of the output shape in order to generate a point cloud from the latent representation. The authors claim that the approach reduced the number of parameters in the decoder to 7 % of the number achieved with conventional fully connected layers.

The PC-AE proposed by Achlioptas et al. in [21] is an example of the most recent architectures, where convolutions substitute fully connected layers and the input data remains in point cloud format. The encoder-part of the network consists of five layers with 1-D convolutions followed by ReLU activation functions, which is in turn followed by a max pooling operation over the calculated features (Figure 1). The unidimensional convolutions allow the network to address each point individually, which, together with the max pooling operation, makes the network order-invariant, overcoming one of the major difficulties in data processing with point clouds.

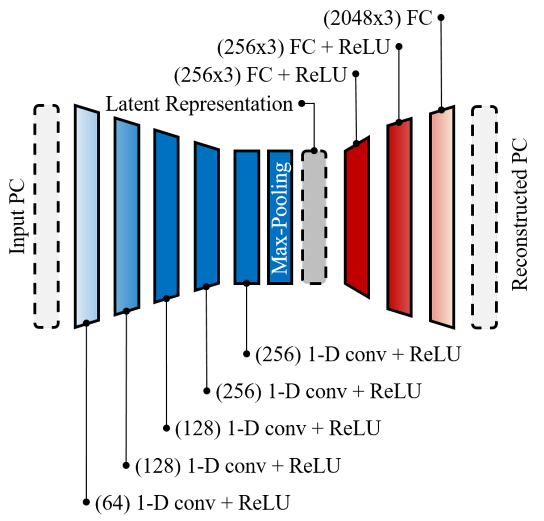

Fig. 1. Representation of the architecture proposed by Achlioptas et al. in [21] with indication of the filter sizes and activation functions.

Analogous to Equation 1, we can calculate the number of parameters in the PC-AE with respect to input size, N_i, and assuming a point cloud defined in 3D Euclidean space as

$$\begin{aligned} N_p^* = &F_{c,1}(3+1) + \sum_{l=2}^{5} \Big[F_{c,l}(F_{c,l-1}+1) \Big] \\ &+ 3\Big[\sum_{l=6}^{7} H_l(H_{l-1}+1) + N_i(H_8+1) \Big], \end{aligned} \tag{2}$$

where $F_{c,l}$ is the number of features in the convolutional filter used in layer l, and H_5, the size of the latent layer. Hence, other than for fully connected layers, the size of the point cloud affects only the number of parameters in the last layer of the decoder, yielding a linear dependency on the size of the output layer.

Due to smaller number in model parameters, the approach proposed in [21] is expected to scale better to larger input sizes than previous PC-AE architectures. However, when increasing the size of input point clouds, it has to be considered that at some point a higer number of points, i.e., a more densily sampled surface, may lead to an oversampling of the shape. In fact, Tenbrinck et al. report in [27] that dense point clouds are prone to represent redundant data. Hence, when increasing the sampling density one has to take care not to include irrelevant or redundant geometric information such as to not unnecessarily increasing computational demand without providing additional geometric information, from which the PC-AE can learn a meaningful latent representation of the input.

Along the lines of avoiding an oversampling of the shape, Gadelha et al. propose a multi-resolution PC-AE [28], where a large base point cloud was downsampled to two different lower-dimensional ones and provided as parallel input to the autoencoder. Throughout the architecture, the convolutions applied to each resolution were dependent, therefore, the features related to fine details contained in the highest-resolution point cloud could be abstracted, while the characteristics related to the positioning and distribution of the geometry over space was enforced by the lower-dimensional representations. Nevertheless, the results achieved for classification and shape reconstruction were comparable to the state-of-the-art [11], [20], while the complexity of the network increased.

An alternative approach to modifying the architecture to handle large inputs, is to change how the point cloud is sampled either from physical objects or from other 3D representations. In particular, a sampling scheme targeted at certain features can be used to adapt the dimensionality of input shapes to PC-AEs available in the literature. In [27] the authors propose a weighted graph-based sparsification method for point clouds, inspired by the Cut Pursuit algorithm [29] and motivated by the sensitivity of the current methods, such as random sampling and tree-based selection, to noise and lack of adaptability. In the paper, the authors claim that the method can be extended to higher dimensional data structures and higher computational efficiency. An interesting aspect of the study in [27] is the use of graph information reducing the dimensionality of the geometric representation. Undirected graphs are widely used for domain representation in engineering applications, where they are usually named as meshes. Hence, for point clouds derived from such representations, graph-based sampling or filtering represents a great potential for pre-reduction of dimensionality in large point clouds, increasing the efficiency of the representation for deep learning tasks.

Chen et al. discuss in in [30] fast resampling methods for point clouds based on graph operators, where the problem is approached from a theoretical signal processing perspective. Their approach aims at achieving an optimum sampling distribution while maintaining certain features contained in the shape. In order to do so, the authors define filters using an approximation of the adjacency matrix, which is used to calculate a metric that indicates the probability for each point of being kept in the representation. Figure 2 shows examples of resampled point clouds using three sampling schemes: random-uniform sampling, as well as low-pass and high-pass filtering as proposed in the paper. The low-pass approach leads to higher number of samples in smooth areas, such as the flat faces in the plate, while the high-pass filter increases the probability of point on edges and abrupt changes in the mesh to be selected.

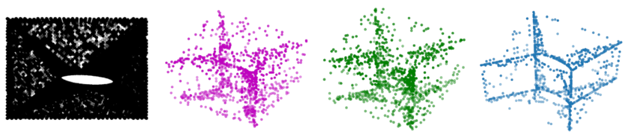
Fig. 2. From the left to the right, top view of a point cloud followed by downsampled representations obtained with random uniform selection, low- and high-pass filters, according to the proposal in [30].

In contrast to graph-based sampling approaches, Öztireli et al. in [31] point out that manifold sampling is essentially a hard problem, since standard signal processing methods are not applicable and there is not a single parameterization for the complete domain. In order to overcome these difficulties and avoid the use of piece- or patch-wise methods, the authors propose sampling techniques based on spectral properties of manifolds: instead of operating on graphs, the method assumes that a set of points with corresponding normal vectors and a defined kernel function are provided, and the essence of the algorithm relies on measuring the relevance of a point to the manifold using the Laplace-Beltrami spectrum, using matrix perturbation theory. According to their research, the spectral characteristics of a manifold are nearly unique, i.e. there are rare cases where different manifolds share the same spectrum, enabling the use of those properties as metrics for comparing manifolds. The quality of the surface reconstruction achieved in the experiments was comparable or better than the methods available in the literature, within reasonable computational effort (e.g. subsampling and surface reconstruction from 14×10^6 to 300×10^3 samples in 6 minutes). However, there is no theoretical guarantee of optimal sampling and all experiments were performed using nearly uniformly sampled geometries. Hence, the reported performance might decrease in the general case.

III. Experimental Set-up for Evaluating PC-AE Training Efficiency and Sampling Strategies

We investigated the scalability of PC-AEs using a two-step approach: first, we investigated the computational demand of the training for increasing point cloud sizes in terms of running time and GPU memory usage. In particular, we determined limits in computing capabilities with respect to the maximum point cloud size that could be processed on current GPU hardware. All tests were performed using a PC-AE, which was adapted from [21] in order to reduce computational demand during training and to improve the ability to learn latent representations for the use in further machine learning applications. Second, we investigated whether through targeted sampling of shapes, a better trade-off between point cloud size and the encoding of relevant geometric information could be achieved. To this end, we applied a high-pass filtering sampling-approach that specifically encoded high-frequency features on the shape. We compared the high-pass filtered sampling to random uniform sampling in terms of reconstruction loss in training and test set. We applied our model to synthetic data where we had full control over the data generation process, such that we could evaluate the learned latent features by correlating them to the known true parameters underlying the generation of the input data set.

All experiments were performed on a machine with two Intel Xecon CPUs, clocked at 2.10 GHz (16 cores, times 2 hyperthreaded), with two Nvidia Quadro RTX 8000 GPUs (48 GB). Each model was trained using a single GPU such that two models could be trained simultaneously per experiment.

A. Dataset generation

To be able to analyze learned latent features, we generated a synthetic data set from a controlled number of parameters, such that the behaviour of the learned latent variables could be related to the true parameters underlying the data generation process. In particular, we investigated the performance of the PC-AE on finer, i.e., high-frequency, features that may be of particular interest in application domains such as CAE. The proposed synthetic dataset was generated from a parameterized 3D shape, based on the model used in [32].

As base shape a thin plate was used, to which an elliptic orifice was added (Figure 3). The orifice was parameterized by coordinates (x_1, x_2) of the ellipsoid's center, the orientation x_3 of the principal axis, and the aspect ratio between the principal and secondary axis, x_4. The range of parameter values was constrained such that the external borders of the plate was preserved for all shapes and any pair of designs could not be symmetric to each other.

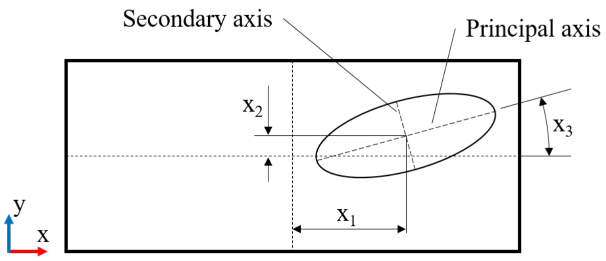

Fig. 3. Parameterization of the base geometry used for generating the dataset.

For the study, 1000 geometries with parameter values drawn randomly from a uniform distribution were generated, using a pipeline of custom Python code, FreeCAD, and Meshlab. Data were generated in FreeCAD as solid and base STL meshes. The latter was then refined in Meshlab to a size of approximately 25000 nodes, which was considered comparable to resolutions found in CAE applications.

B. Point cloud sampling

As a baseline for all experiments, we used random-uniform sampling (RUS), which is unbiased, does not rely on domain-specific assumptions, and can be easily implemented in several applications. We compared RUS to sampling using the graph-based high-pass filter (HPF) proposed in [30]. The filter was applyed to the vertices of the refined mesh, where we obtain the filter-response of the i^{th} point in the graph vertex as

$$(h_{HH}(A)X)_i = \mathbf{x}_i - \sum_{j \in \mathcal{N}_i} A_{i,j}\mathbf{x}_j, \tag{3}$$

where $\mathcal{N}$ represents the point cloud domain, $\mathbf{x}_i$ the vector of the i^{th} point's coordinates, and A is the transition matrix,

$$A = D^{-1}W \tag{4}$$

where W is the symmetric adjacency matrix that representing connectivity between nodes, and D is the diagonal degree matrix, obtained from the sum of the columns of W. When using the transition matrix, the filter reflects how much information is known about a point based on its neighborhood, and it is furthermore shift- and rotation-invariant.

The optimal resampling distribution π^* is proportional to the response of the graph to the filter,

$$\pi^* = \frac{(h_{HH}(A)X)}{\|(h_{HH}(A)X)\|_1} \tag{5}$$

which is the vector of the graph response normalized by its ℓ_1 norm, such that the components sum to 1. Components of higher probability are associated with points close to abrupt changes in the geometries, e.g. edges, while components of lower probability are associated with smoother surfaces.

In [30], the authors assume that typically only the point cloud is known, but not the adjacency matrix, W, which has to be estimated. This step is omitted in the present work since point clouds are extracted from STL meshes, for which the connectivity between points is known. The proposed modification was verified through visual inspection of re-sampled point clouds from geometries randomly selected from the generated dataset (Figure 4).

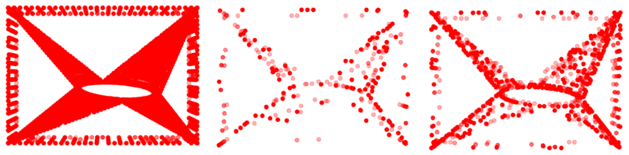
Fig. 4. Point cloud resampling according to the modified high-pass filtering approach proposed in [30]. From the left to the right, full point cloud (25,000 points), 256-point and 2048-point representation.

C. Point cloud autoencoder architecture and training

The PC-AE implemented for the experiments is based on the architecture presented in [21]. The architecture was modified by replacing the ReLU activation functions in the last convolutional layer of the encoder by hyperbolic tangent functions, and the ReLU functions in the last decoder layer by sigmoid functions. We chose the sigmoid function such that the range of the activation function matched the normalized coordinate values of the shapes in the dataset. For the activation functions in the latent layer, we chose the hyperbolic tangent to limit the values of the latent variables such as to increase the interpretability of the latent variables.

Further modifications were performed on the training procedure. As discussed above, the loss function is one of the potential bottlenecks for training large network architectures. Therefore, the training was divided into two parts, where in the first part, a computationally cheaper coarse training was performed by pre-ordering the input point cloud and using the mean square distance between points as loss function under the assumption that the order of the points was the same for input and output. Thus, the costly search part of the Chamfer Distance (CD) was avoided [25]. In the second part of the training, indicated by a stagnation of the loss function, the conventional CD approach was used. The CD algorithm used in the experiments was implemented according to [21] and the epoch when the loss function changed its behavior was defined based on observations made during tests with the dataset and architecture, but kept the same for all the experiments.

In order to impose an ordering on the point clouds, a data partitioning tree algorithm was adopted as a pre-processing step [28]: the point cloud was recursively divided into two parts according to the mean value of the points' coordinates, for each axis and partitioning. Hence, the points were organized as a list of patches which should be consistent among all geometries in the dataset. A disadvantage of the method is that the size of the point clouds becomes constrained to the power of the number of partitions, which in this case was set to two.

In order to verify effects of the modification in the architecture, a model was trained on the car class from ShapeNet Core [33] and compared to the performance reported in [21]. In total, 6350 shapes in batches of 50 were used for training and 375 shapes were used for testing. Parameters were fitted using the Adam optimizer [34] with learning rate of 5×10^{-4}, $\beta_1 = 0.9$ and $\beta_2 = 0.99$ over 500 generations. The switch to the CD was performed after 200 iterations, where this value was identified manually. The reconstruction loss obtained with the proposed architecture was comparable to the results reported in [21] (Table I). We therefore considered the architecture feasible for the experiments performed.

TABLE I
RECONSTRUCTION LOSS OF THE PC-AE TRAINED ON THE CAR CLASS FROM SHAPNET CORE

	Reference	Modified Architecture	Modified Loss Function
$CD_{training}$	3.34E-04	2.91E-04	2.96E-04
CD_{test}	(4.00 ± 0.9)E-04	(3.03 ± 0.8)E-04	(3.08 ± 0.8)E-04

When training the model with the mean squared distance, the reconstructed point clouds were arranged in a grid-like structure, which could be observed in all models. After 100 iterations using the CD, points started to diverge from the grid and spread over the surface of the shape (Figure 5). The grid-like ordering could be an effect of the partitioning algorithm dividing samples into patches.

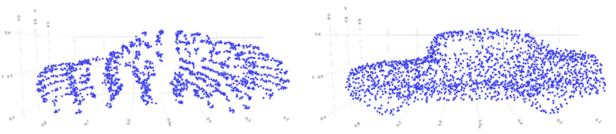

Fig. 5. Shape reconstruction for a sample in the dataset. From the left to the right, the shapes were retrieved after 200 and 300 iterations, respectively.

D. Experiments

We ran experiments in order to evaluate RUS compared to sampling with the HPF in terms of running time of the training, as well as reconstruction loss. We varied the input point cloud size between, 2048, 4096, and 8192 points, where 2048 is the input size commonly used in the literature (e.g., [11], [20], [21]). We furthermore analyzed 8-, 16- and 32-dimensional latent representations, using multiples of four, i.e., the number of parameters underlying the generation of the input data set. Most of the training hyperparameters were kept the same as in [21], except for the number of iterations, which was increased to 8000. We furthermore applied data augmentation by generating three random rotations around the z-axis for each geometry [21], with rotations within the interval [-$\pi/2$, $\pi/2$].

In order to analyze the maximum point cloud size that could be processed with the given hardware and PC-AE, we performed a run-to-crash test by systematically increasing the size of the point clouds with redundant data.

Finally, the analysis of learned or abstracted features was performed in two steps: First, models were evaluated using shapes from different datasets and different sampling schemes, in order to identify overfitting and potential for generalization. In a second step, the models were verified by visually comparing the reconstructed point clouds with their respective inputs, as well as by performing interpolations within the learned latent variables.

IV. Results and Discussion

A. Running time for PC-AE training

The running time of the PC-AE training in terms of time elapsed per epoch was more sensitive to the size of the point cloud than to the dimensionality of the latent space, as was expected according to eqs. 1 and 2 (Table II). Running times were approximately equal for both sampling methods since the pre-processing stage was not taken into account.

B. Evaluation of maximum input size

In order to asses the maximum point cloud size that could be processed using the available hardware and architecture, a run-to-crash test was performed. In this experiment, the size of the point clouds was gradually increased by replicating points, and the algorithm performed an attempt to start the training process for 20 iterations. The maximum number of samples achieved was 131,072 (2^{17}), with an average GPU memory usage of (56.49 $\pm$ 14.58) %, for a single graphic card, and elapsed time for 10 iterations over 260s. If the data partitioning tree was not used, the size of the point clouds can be increased to 200,000 points, achieving (45.38 $\pm$ 19.91) % of memory usage, on average, and elapsed time per 10 iterations of 450s.

TABLE II
TIME AND MEMORY REQUIREMENTS FOR DIFFERENT POINT CLOUD (PC) AND LATENT REPRESENTATION (LR) SIZES.

LR size	PC	GPU memory usage	$\Delta t_{10epochs}$
8	2048	(15.89 $\pm$ 0.99)%	9s
	4096	(28.14 $\pm$ 2.04)%	11s
	8192	(40.12 $\pm$ 3.25)%	17s
16	2048	(17 $\pm$ 1.28)%	10s
	4096	(30.86 $\pm$ 3.65)%	10s
	8192	(44.15 $\pm$ 4.74)%	15s
32	2048	(17.85 $\pm$ 1.16)%	11s
	4096	(31.77 $\pm$ 2.61)%	11s
	8192	(47.97 $\pm$ 3.54)%	16s

C. Reconstruction quality for RUS and HPF sampling schemes

We compared the reconstruction error from a PC-AE trained on data sampled either using RUS or using HPF and found that both sampling schemes had losses of comparable order of magnitude (Table III). However, networks trained on geometries sampled with HPF performed better during training, while the RUS led to better performance on unseen data. The overall best performance on test data for point clouds of size 8192 was achieved using a latent space size of 8 and the RUS.

In order to verify the capacity of the network to generalize to unseen data, the models were tested on 500 shapes sampled with the opposing approach, i.e. networks trained on RUS-sampled geometries were tested on HPF data, and *vice versa* (Table IV, where $CD_{a,b}$ indicates training on dataset a and testing on dataset b). Results indicate that models trained using the RUS dataset tended to handle unseen data better, except for increasing sizes of point clouds and latent representations. The behavior observed in the experiment may be explained by the variety of information contained in the dataset, which is expected to be higher with the RUS approach, since the method is not feature-aware and the sampling follows a uniform distribution. Hence, when the number of samples increases, the HPF approach starts to select points less relevant with respect to changes in the geometry, approximating itself to the RUS method, and increasing generalization-ability. An increase in the dimensionality of the latent space furthermore means increasing the number of parameters in the network and, therefore, its ability to abstract data, which is also in line with the observed results.

D. Evaluation of learned latent representations

To illustrate that training a network on point clouds applying HPF learned different features from training the network using RUS, we show the reconstruction of an exemplary shape, once for the sampling scheme for the example shape and matching the sampling in the training data set, and once for the sampling schemes not matching (Figure 6). When comparing the reconstruction (red) to the reference point clouds (blue), it

TABLE III
CHAMFER DISTANCE (CD) CALCULATED ON THE TRAINING AND TEST SETS SAMPLED ACCORDING TO THE RUS APPROACH, FOR DIFFERENT POINT CLOUD (PC) AND LATENT REPRESENTATION (LR) SIZES.

		$CD_{training}$		CD_{test}	
LR size	PC size	RUS	HPF	RUS	HPF
8	2048	1.18E-04	**5.12E-05**	(**1.95** ± 0.20)E-04	(2.15 ± 0.67)E-04
	4096	6.33E-05	**3.29E-05**	(**1.20** ± 0.12)E-04	(1.39 ± 0.21)E-04
	8192	4.01E-05	**3.37E-05**	(**9.29** ± 0.10)E-05	(12.9 ± 1.7)E-05
16	2048	9.45E-05	**3.02E-05**	(1.89 ± 0.25)E-04	(**1.44** ± 0.22)E-04
	4096	4.76E-05	**2.54E-05**	(**1.13** ± 0.05)E-04	(1.37 ± 0.14)E-04
	8192	3.93E-05	**1.32E-05**	(9.58 ± 0.01)E-04	(**1.23** ± 0.15)E-04
32	2048	8.16E-05	**2.64E-05**	(1.73 ± 0.21)E-04	(**1.25** ± 0.11)E-04
	4096	4.27E-05	**1.60E-05**	(**1.03** ± 0.06)E-04	(1.17 ± 0.11)E-04
	8192	2.96E-05	**1.15E-05**	(**1.18** ± 0.18)E-04	(1.23 ± 0.15)E-04

TABLE IV
CHAMFER DISTANCE OBTAINED FROM TESTS ON DATASETS GENERATED WITH DIFFERENT SAMPLING TECHNIQUES.

LR size	PC size	$CD_{RUS,HPF}$	$CD_{HPF,RUS}$
8	2048	(**4.28** ± 0.56)E-04	(5.72 ± 1.50)E-04
	4096	(**3.55** ± 0.48)E-04	(4.10 ± 1.30)E-04
	8192	(**3.98** ± 0.52)E-04	(4.44 ± 1.00)E-04
16	2048	(**4.07** ± 0.92)E-04	(5.87 ± 1.60)E-04
	4096	(3.92 ± 0.66)E-04	(**3.60** ± 0.96)E-04
	8192	(**3.91** ± 0.49)E-04	(11.7 ± 1.4)E-04
32	2048	(4.07 ± 0.96)E-04	(**4.00** ± 0.49)E-04
	4096	(3.92 ± 0.66)E-04	(**3.46** ± 1.10)E-04
	8192	(3.91 ± 0.49)E-04	(**3.45** ± 0.41)E-04

can be seen that both models provide a good approximation of the geometry when the sampling scheme matches the one used in the training data set and that this is no longer the case for a mismatch in sampling schemes. Furthermore, the PC-AE trained on RUS training data lead to a visullay better approximation of the input than the HPF example, matching the quantitative results reported above.

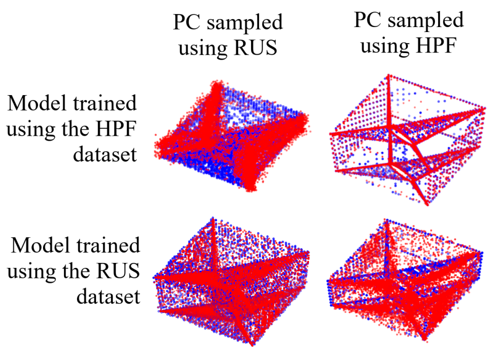
PC sampled using RUS
PC sampled using HPF
Model trained using the HPF dataset
Model trained using the RUS dataset

Fig. 6. Reconstruction of geometries using the architectures trained on different datasets. The blue markers indicate the points of the input point cloud.

As a second visual verification, we interpolated between two shapes in the latent space, using a PC-AE trained on different datasets (see Figure 7 for an example). Regardless of the accuracy in the reconstruction, the trained models were capable of smoothly interpolating between several different shapes, indicating that the latent representation was able to successfully abstract and represent the geometric features.

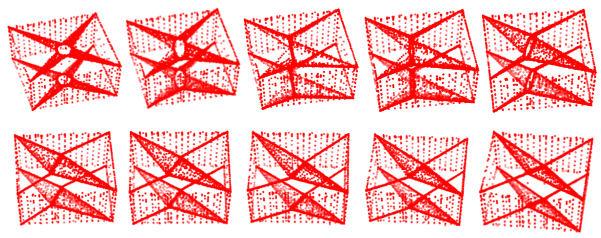

Fig. 7. Interpolation between geometries using the dataset sampled with the HPF method. The progression goes from left to right, top to bottom.

We further verified that the PC-AE indeed learned the latent features underlying the used synthetic data set by calculating the Pearson correlation coefficient between design variables (DVs), x_i, and learned latent variables (LV) on 150 shapes from each dataset. As a validation, we calculated the pair-wise correlation between DVs to assert that design variables were indeed independent, where the highest magnitude among the coefficients was 0.24 between variables x_2 and x_3, and all other correlations were close to zero.

Next, correlations between LVs and DVs were calculated. For simplicity, only the models with eight latent variables and trained on the datasets with 8096 points were considered for the PC-AE trained on the RUS and HPF datasets, respectively (Figure 8). Based on the similar magnitude of correlations found in both autoencoders, we conclude that both sampling schemes lead to an equal ability of the PC-AE to abstracted the features of the dataset. The model trained on the RUS dataset presented a higher correlations overall.

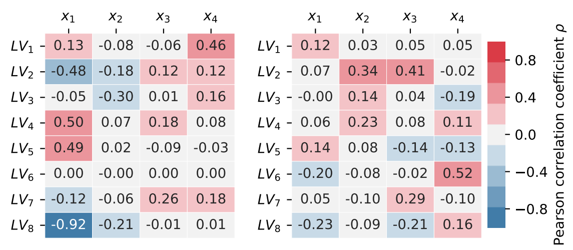

Fig. 8. Pearson correlation between the design variables x_i and latent variables (LV) obtained from the network trained on the RUS (left) and HPF dataset (right).

Finally, when analyzing the pair-wise correlation between LVs for both models, the HPF method yielded less correlated variables than the RUS (Figure 9). Potentially, HPF lead to a more homogeneous problem and hence to more uniform features in the HPF dataset across models. On the other hand, when RUS is used, the process of sampling the points is independent for each model such that the models are less similar. Therefore, for RUS, abstracting the features used to generated the dataset becomes harder and the distributions of points in the models are mapped differently. This is in line with the superior, generalization capabilities observed for PC-AE models trained on RUS data.

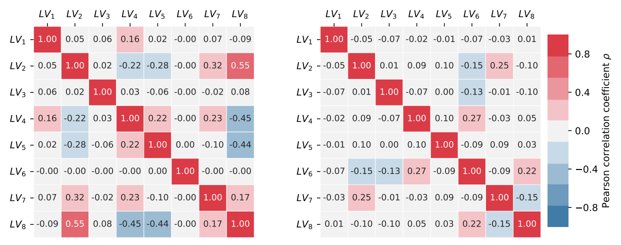

Fig. 9. Pearson correlation between latent variables (LV) obtained from the network trained on the RUS (left) and HPF dataset (right).

V. Conclusion

The research on GDL has advanced considerably in the recent years and the boost provided by the development of powerful processing units such as GPUs played an important role in this process. Among the data representations used for GDL, point clouds have become a promising approach due to their simple structure, efficiency, and potential to resolve 3D objects with high resolution, which is required for an application in the automotive digital development. The present work investigated the scalability of PC-AE from two perspectives.

First, we proposed a modification in the loss function, in order to avoid using point-wise operations during the complete training phase, which is one of the bottlenecks for using large point cloud models. Our approach was tested on the car class from ShapeNet Core [33] and achieved comparable performance to the architecture proposed in [21]. The performance of the PC-AE in the experiments was evaluated in terms of computational effort and abstraction of features. In terms of computational effort, our approach could be scaled up to 200,000 points, two orders of magnitude greater than the models used in most of the reviewed works. Comparing the runtime and memory usage for different combinations of sizes of the latent representation and the point cloud. The latter had a higher influence on the computational costs, due to the resulting increase in units in the input layer, leading to greater increase in model parameters than increasing the size of the latent layer.

Second, we proposed the reduction of the dimensionality of the point clouds by a sampling scheme based on feature detection (HPF). In order to assess the effects of the sampling scheme on the learned features, a synthetic dataset was generated allowing for full control over the data generation process in terms of known number and types of parameters, as well as their values. This dataset allowed for the straightforward evaluation of learned features through correlation of latent space actications and design parameters. When comparing sampling HPF to random uniform sampling, achieved reconstruction losses indicated that models trained on randomly sampled geometries generalized better than the ones trained on the dataset sampled using a HPF, since they had more information about the overall geometry while the HPF focused on the detection of edges and vertices.

Finally, the latent variables of the PC-AE trained on the RUS dataset showed a stronger correlation to the design parameters, which supports the good generalization capability of the model. However, the model trained on the HPF sampled dataset showed weaker correlation within the latent space, which was a possible sign that the sampling method enabled the autoencoder to differentiate the geometric features more efficiently. Future work should investigate if tasks other than reconstruction, as solved by the autoencoder, benefit from the proposed preprocessing through feature detection.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 766186.

References

[1] P. Wang, Y. Gan, Y. Zhang, and P. Shui, "3d shape segmentation via shape fully convolutional networks," *CoRR*, vol. abs/1702.08675, 2017.
[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016.
[3] Z. Huang, C. Wan, T. Probst, and L. V. Gool, "Deep learning on lie groups for skeleton-based action recognition," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 1243–1252.
[4] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris, "Deep Learning Advances in Computer Vision with 3D Data," *ACM Computing Surveys*, vol. 50, no. 2, pp. 1–38, 2017.
[5] E. Ahmed, A. Saint, A. E. R. Shabayek, K. Cherenkova, R. Das, G. Gusev, D. Aouada, and B. Ottersten, "Deep learning advances on different 3D data representations: a survey," *ArXiv preprint*, pp. 1–20, 2018.
[6] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. July, pp. 18–42, 2017.
[7] M. Tatarchenko, A. Dosovitsky, and T. Brox, "Octree Generating Networks : Efficient Convolutional Architectures for High-resolution 3D Outputs," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2088–2096.
[8] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *Proceedings of the IEEE international conference on computer vision workshops*, 2015, pp. 37–45.
[9] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5425–5434.
[10] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black, "Generating 3D faces using Convolutional Mesh Autoencoders," in *15th European Conference on Computer Vision (ECCV)*, vol. arxiv:1807, 2018.

[11] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017.

[12] T. Friedrich, N. Aulig, and S. Menzel, “On the Potential and Challenges of Neural Style Transfer for Three-Dimensional Shape Data,” in *International Conference on Engineering Optimization*. Springer International Publishing, 2019, pp. 581–592.

[13] H. F. Murcia, M. F. Monroy, and L. F. Mora, “3d scene reconstruction based on a 2d moving lidar,” in *Applied Informatics*, H. Florez, C. Diaz, and J. Chavarriaga, Eds. Cham: Springer International Publishing, 2018, pp. 295–308.

[14] Z. Ouyang, Y. Liu, C. Zhang, and J. Niu, “A cgans-based scene reconstruction model using lidar point cloud,” in *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, Dec 2017, pp. 1107–1114.

[15] Y. Zhong, S. Wang, S. Xie, Z. Cao, K. Jiang, and D. Yang, “3d scene reconstruction with sparse lidar data and monocular image in single frame,” *SAE Int. J. Passeng. Cars ? Electron. Electr. Syst.*, vol. 11, pp. 48–56, 09 2017.

[16] W. Zhang, H. Jiang, Z. Yang, S. Yamakawa, K. Shimada, and L. B. Kara, “Data-driven upsampling of point clouds,” *Computer-Aided Design*, vol. 112, pp. 1 – 13, 2019.

[17] C. Duan, S. Chen, and J. Kovacevic, “Weighted multi-projection: 3d point cloud denoising with tangent planes,” in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Nov 2018, pp. 725–729.

[18] J. Zeng, G. Cheung, M. Ng, J. Pang, and C. Yang, “3d point cloud denoising using graph laplacian regularization of a low dimensional manifold model,” *CoRR*, vol. abs/1803.07252, 2018.

[19] V. Shapiro, “Chapter 20 - solid modeling,” in *Handbook of Computer Aided Geometric Design*, G. Farin, J. Hoschek, and M.-S. Kim, Eds. Amsterdam: North-Holland, 2002, pp. 473 – 518.

[20] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep hierarchical feature learning on point sets in a metric space,” in *NIPS*, 2017.

[21] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. J. Guibas, “Representation learning and adversarial generation of 3d point clouds,” *CoRR*, vol. abs/1707.02392, 2017.

[22] Z. Nie, H. Jiang, and L. B. Kara, “Deep learning for stress field prediction using convolutional neural networks,” *arXiv preprint arXiv:1808.08914 [cs.LG]*, 2018.

[23] M. Tschannen, O. Bachem, and M. Lucic, “Recent advances in autoencoder-based representation learning,” in *Workshop on Bayesian Deep Learning (NeurIPS 2018)*, no. NeurIPS, 2018, pp. 1–26.

[24] P. Mandikal and R. V. Babu, “Dense 3d point cloud reconstruction using a deep pyramid network,” *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1052–1060, 2019.

[25] H. Fan, H. Su, and L. Guibas, “A point set generation network for 3D object reconstruction from a single image,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 2463–2471, 2017.

[26] Y. Yang, C. Feng, Y. Shen, and D. Tian, “FoldingNet: Point cloud autoencoder via deep grid deformation,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 206–215, 2018.

[27] D. Tenbrinck, F. Gaede, and M. Burger, “Variational graph methods for efficient point cloud sparsification,” *CoRR*, vol. abs/1903.02858, 2019.

[28] M. Gadelha, R. Wang, and S. Maji, “Multiresolution tree networks for 3d point cloud processing,” in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 105–122.

[29] L. Landrieu and G. Obozinski, “Cut Pursuit: fast algorithms to learn piecewise constant functions on general weighted graphs,” *SIAM Journal on Imaging Sciences*, vol. Vol. 10, no. No. 4, pp. pp. 1724–1766, 2017.

[30] S. Chen, D. Tian, C. Feng, A. Vetro, and J. Kova?evi?, “Fast resampling of three-dimensional point clouds via graphs,” *IEEE Transactions on Signal Processing*, vol. 66, no. 3, pp. 666–681, Feb 2018.

[31] A. C. Öztireli, M. Alexa, and M. Gross, “Spectral sampling of manifolds,” *ACM Trans. Graph.*, vol. 29, no. 6, pp. 168:1–168:8, Dec. 2010.

[32] S. Ladjal, A. Newson, and C. Pham, “A PCA-like autoencoder,” *CoRR*, vol. abs/1904.01277, 2019.

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

[34] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *International Conference on Learning Representations*, 12 2014.