Evaluation of Latent Space Learning with Procedurally-Generated Datasets of Shapes

Sharjeel Ali Oliver van Kaick
School of Computer Science, Carleton University, Canada
{sharjeel.ali,oliver.vankaick}@carleton.ca

Abstract

We compare the quality of latent spaces learned by different neural network models for organizing collections of 3D shapes. To accomplish this goal, our first contribution is to introduce a synthetic dataset of shapes with known semantic attributes. We use a procedural method to generate a dataset comprising four categories, with a total of over 10,000 shapes, providing a controlled setting for studying the properties of latent spaces. In contrast to previous work, the synthetic shapes generated with our method have a more realistic appearance, similar to objects in manually-modeled collections. We use 8,800 shapes from the generated dataset to perform a quantitative and qualitative evaluation of the latent spaces learned with a set of representative neural network models. Our second contribution is to perform the quantitative evaluation with measures that we developed for numerically assessing the properties of the latent spaces, which allow us to objectively compare different models based on statistics computed on large sets of shapes.

1. Introduction

Recently, a variety of methods have been introduced for synthesizing 3D shapes with deep neural networks [4]. Most of these methods are based on the idea of learning a latent space that organizes an existing collection of 3D shapes and allows the generation of new shapes. Specifically, each shape is represented as a low-dimensional latent representation, e.g., a small vector of numbers. Then, a decoder neural network can decode the latent vector back into a 3D shape with rich detail. In addition, the latent vectors can be interpolated/extrapolated and then decoded into new shape variations, allowing users to explore the shape space at a high-level by manipulating the latent vectors. These methods have attracted much interest in the literature since the latent spaces can be learned without pre-defining features or computing correspondences among the shapes.

Different types of deep network architectures can be used for learning latent spaces of shapes, such as autoencoders (AEs) [1, 11, 6, 5], variational autoencoders (VAEs) [17], auto-decoders [18], and generative adversarial networks (GANs) [22, 1, 6]. AEs and VAEs allow to explicitly encode shapes into the latent representation. GANs implicitly learn a latent space, where the input noise vector can be seen as a latent encoding, although arbitrary shapes cannot be directly mapped to this space; the latent space has to be explored in an indirect manner. Thus, GANs have been extended to also include an encoder, such as in the BiGAN [8] and VAE-GAN [15] models used for image synthesis. While most previous work on shape synthesis with deep networks has focused on improving the visual quality of the synthesized shapes by using different shape representations and network architectures, few works have studied the capabilities and limitations of the latent spaces learned.

In this work, our goal is to investigate the quality of the latent spaces learned by existing deep neural networks for 3D shapes and their potential for shape modeling. Specifically, our goal is to study the quality of the organization of shapes in the latent spaces and also the quality of interpolated shapes. A meaningful organization of the latent space would imply that certain directions in this space capture the variation of particular semantic attributes of the shapes, while good interpolation capabilities would imply that the generated shapes are smooth and plausible variations of the training shapes [8]. Although these properties are not explicitly optimized for in most neural networks, they are often seen as an expected output of the learning process, given that these properties are evaluated in previous works [22].

To perform this investigation, our first contribution is to introduce a synthetic dataset of shapes with known structure and semantic attributes (Figure 1). We use a procedural method [19] to generate a large dataset where the shapes gradually differ in their semantic attributes and corresponding visual appearance. The datasets generated with our method then serve as a controlled setting for our study of latent spaces. In contrast to previous work [14], the shapes generated with our method have a more realistic appearance, rather than being just a collection of boxes or proxies.

Figure 1: A gallery of 40 shapes sampled from our 4 procedurally-generated datasets.

We introduce a dataset with four categories of shapes, which we use to evaluate how well the latent spaces learned by different variations of autoencoders organize the semantic attributes of the shapes. We focus on networks that learn the embeddings from non-structured 3D content, such as volumes, rather than structured content [4]. We use non-structured representations as they require minimal data preparation, compared to structured content that is often represented in the form of graphs of parts with labels.

Moreover, our second contribution is to introduce a set of measures to evaluate the learned latent spaces in a quantitative manner. Specifically, we measure the distance correlation of a parametric representation of the shapes with the latent vectors, to compare the shape attributes with the shape organization in the latent space. Secondly, we examine how well the latent space directions organize the shape attributes by measuring how well the latent dimensions group specific shape attributes. Finally, we also evaluate the interpolation capabilities of the networks by measuring the similarity between interpolated and expected shapes.

In summary, our contributions are:

- Four synthetic datasets of 3D shapes with known semantic attributes, which can be used in controlled experiments or for learning in general;
- Three procedures for evaluating how well learned latent spaces conform to the semantic attributes of shapes;
- An evaluation of the latent embeddings learned by four variations of autoencoders;

2. Related work

In this section, we discuss the previous work most related to our study: learning of latent spaces for encoding and synthesizing 3D shapes, and the evaluation of such spaces.

Encoding of 3D shapes. A variety of deep network architectures have been proposed for learning latent spaces, such as AEs, VAEs, and GANs. These architectures have also been applied to 3D shapes, based on different shape representations. The simplest type of AE has a symmetric architecture, where an input shape is encoded into a latent representation and then decoded back into the same type of shape representation. For example, Achlioptas et al. [1] use a symmetric AE to encode shapes represented as point clouds. However, hybrid architectures are also possible, where the encoder and decoder use different representations. In practice, this can be achieved by first training the encoder with a symmetric decoder, and then exchanging the decoder for a network generating a shape in a different type of representation. For example, Chen and Zhang [6] use an AE for learning a latent space of shapes, where the encoder can process different types of input while the decoder generates shapes represented as implicit occupancy fields.

In addition, Groueix et al. [11] use an AE to synthesize a shape as a set of parametric surface patches. Park et al. [18] introduce a method for shape generation based on a signed distance implicit function and auto-decoder network, which does not require the use of an encoder network. Chibane et al. [7] propose the use of an unsigned distance function to represent open surfaces and objects with internal structures.

Moreover, GANs have improved the visual quality of synthesized data with the use of adversarial learning, and thus this type of method has also been applied to the synthesis of 3D shapes. For example, Wu et al. [22] generate shapes with a GAN based on a volumetric representation. The works of Achlioptas et al. [1] and Chen and Zhang [6] discussed above also experimented with the use

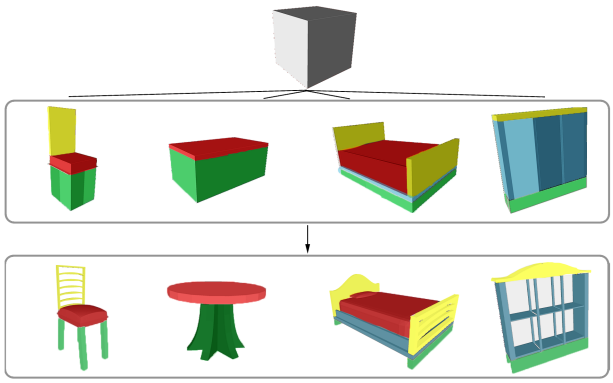


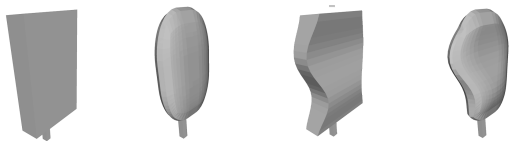
Figure 2: Procedural generation of man-made shapes with split grammars. An initial box is refined recursively with a set of rules into a set of boxes that define the fine structure of a shape. The final boxes are then refined with patches and deformed to define the final meshes of the shapes.

of GANs for shape generation, achieving smoother visual results. Chen et al. [5] extend the use of an implicit representation to generate shapes with binary space partitioning.

Note that, since standard GANs do not train an encoder, the latent spaces are implicitly defined and cannot be explicitly explored. Thus, a few methods have been developed for adding an encoder network to a GAN and using it in image synthesis tasks [8, 15]. However, these encoder GANs have not been extended yet to synthesize 3D shapes.

Evaluation of latent spaces. An important question when considering different models for learning latent spaces is how to evaluate and compare such models. Thus, a few methods have been proposed in the machine learning literature for this task. One of the earliest forms of evaluation proposed is to show a 2D embedding of the learned representation where each data point is colored with its class label [13]. This method enables a visual inspection of the grouping of points in the 2D latent space, but is only applicable to datasets with a single label per point.

Theis et al. [20] discuss different methods for the evaluation of generative models, such as Parzen window estimates and visualizing samples and nearest neighbors to assess the generalization capabilities of the model. The inspection of samples and nearest neighbors allows to detect overfitting to some extent when appropriate metrics are used, but the evaluation is mainly qualitative. Burgess et al. [3] evaluate the disentanglement of learned image representations by creating a dataset of Gaussian blobs with known positions, and then verifying how well the latent spaces are organized according to the blob parameters. We follow a similar idea by creating a dataset with known properties, although we focus on 3D shapes and create more complex and realistic data.



(a) box (b) round (c) deform (d) b+c

Figure 3: Examples of rounding and deformation rules. Starting with a box (a), the method can apply a rounding (b) or deformation (c) function. Both rules can then be combined to create a rounded and deformed shape (d).

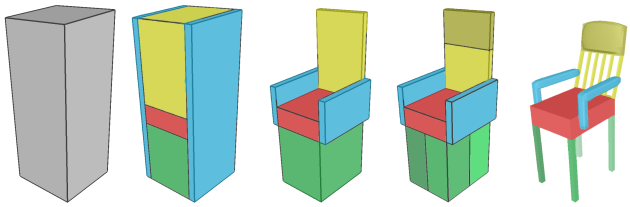


Figure 4: Example of a chair generated with our procedural method. The figure shows the different levels of recursive refinement of the boxes and the final generated shape.

3. Procedural generation of man-made shapes

Overview of the procedural modeling system. In this section, we provide some background on the procedural system to explain how we generated the shapes in our datasets. Different methods have been proposed for the procedural generation of shapes [16, 19]. In our work, we generate man-made shapes in a procedural manner with a method inspired by the *split grammar* of Wonka et al. [21], which was originally introduced for the procedural generation of buildings. The idea of the split grammar is to recursively split pieces of geometry into smaller units based on a set of rules. For the generation of man-made shapes, our idea is to define rules that operate at the *box* and *patch* level, as shown in Figure 2. That is, given an initial box enclosing the entire space where the shape will reside, the box is recursively split into smaller boxes to define the fine structure of the shape. Then, the resulting boxes are transformed into a set of surface patches that define the mesh geometry of the shape. Finally, the patches can undergo deformation to enable the creation of curved surfaces as shown in Figure 3.

The use of boxes and patches allows us to generate complex shapes with detailed geometry, as opposed to shapes represented only by cuboids, and allows us to generate shapes without involving expensive operations such as constructive solid geometry operations or surface reconstruction. The box splitting, patch creation and deformation are all carried out with a set of rules. We loosely call the set of all rules that generate shapes of a specific category a *gram-*

mar. A rule takes as input one piece of geometry (box or patch) and outputs a set of refined geometry (one or more boxes or patches). Thus, the implementation of rules is not complicated as it involves considering only a small amount of information at a time. Figure 4 shows an example of the recursive splitting of boxes and the final shape created.

In addition to its geometry, a box or patch can be assigned a set of tags, which are considered by the rules during the refinement of the shape. Tags can be used to attach semantic meaning to the shape parts. For example, the box on top of a future chair can be assigned the tag "back", which then ensures that only rules that create chair backs are applied to this box. By defining multiple rules that accept the same sets of tags, we can add stochasticity into the system and generate a large variety of shapes.

Shape generation. The method for recursive application of split rules uses a list structure to store all active geometry, i.e., boxes or patches that can still be further refined. The list is initialized with a box enclosing the allowed work space. The main execution step of the method involves iteratively taking a piece of geometry G from the list and applying a rule that: 1. Handles the type of G (box or patch); 2. Requires the same tags that G has been assigned. If multiple rules can be applied, we randomly select a rule to generate stochastic shape variations. The newly generated geometry is appended to the list if it can be further refined. The method terminates when the list is empty.

4. Generated datasets

Shape parameters. The rules in the shape grammar control different aspects of the shape generation, such as the presence of certain types of parts or the style and visual look of the parts. Before creating the datasets in our study, we first associate a set of parameters with all of the grammar rules for a category, so that a shape can be encoded as a parameter vector. We consider three types of parameters: binary (encoding whether a shape feature exists or not), integer (a feature is of a certain type denoted by a finite set of integer numbers), or scalar (a continuous varying feature such as the height of a part, sampled into a set of s steps). In order to accurately compare vectors with differing integer parameters, we convert each integer parameter with a set of t types into t binary parameters, in which only the parameter corresponding to a given integer is set to 1.

Shape generation. To generate the shapes in our datasets, we systematically create all the possible combinations of parameters and generate the corresponding shapes. In this manner, we create four datasets. We list the datasets and their statistics on Table 1. Examples of shapes in the datasets are shown in Figure 1. Then, we stochastically sample a subset of 2,200 shapes from each category for our study, to arrive at datasets that fit within the memory limits of the hardware used for training the different models. The final datasets are then given by the sampled shapes and their associated parameter vectors.

Dataset	# shapes	# parameters			# parts	# steps
		B	S	I		
Beds	2,430	0	5	3	1	3-6
Chairs	2,592	0	3	13	4	3
Tables	2,880	0	4	6	2	3-5
Shelves	2,916	2	5	3	3	3

Table 1: Characteristics of the datasets generated with the procedural method for our study. We report the number of shapes (# shapes), number of binary (B), scalar (S), and integer (I) parameters, number of semantic parts (# parts), and the number of steps (# steps) sampled per scalar parameter.

5. Shape encoding architectures

Since GAN architectures combined with encoders have not yet been extended to 3D shapes, in our evaluation, we consider mainly different variations of autoencoders. We evaluate the following network architectures.

3D-AE. This is the classic autoencoder architecture adapted to 3D volumes of shapes. We use the implementation provided by Guan et al. [12], where the network takes as input a 32^3 voxelized shape that is encoded through 4 convolutional layers into a 128-dimensional vector. The model is trained with the *mean squared error* loss function.

3D-VAE. This is an implementation of the 3D variational autoencoder introduced by Brock et al. [2], with an architecture similar to the 3D-AE. The model is trained with the *KL divergence* loss.

3D-GAE. We also consider the 3D generalized autoencoder (GAE) introduced by Guan et al. [12]. Their architecture takes as input a 32^3 volume and encodes it into a 128-dimensional latent vector through 3 convolutional layers. The GAE learns to reconstruct a set of instances rather than just a single instance from a given training sample. The set of instances is defined by the training sample and its nearest neighbors (NNs), where the NNs are determined by the Chamfer distance computed on the shape volumes [10].

3D-PGAE. We modify the original 3D-GAE model by changing the method for determining the NNs. Instead of using the Chamfer distances computed on the voxelized shapes, we use the parameter vectors to determine the NNs.

This constrains the learning of the latent space according to the parameter attributes and represents a method that is given prior knowledge about the shape similarity for learning the latent space.

6. Evaluation procedures

In this section, we discuss the evaluation procedures that we introduce for evaluating the learning of latent spaces. Our first goal is to study the quality of the organization of shapes in the learned latent spaces. A meaningful organization of a latent space implies that the space accurately captures the variation of semantic attributes of the shapes, which is reflected in at least two properties: (i) Shapes that are similar in terms of their semantic attributes should be close together in the latent space; and (ii) Certain directions in this space should capture the variation of particular semantic attributes of the shapes. Moreover, our second goal is to evaluate the quality of interpolated shapes, i.e., shapes generated by walking in the latent space. Good interpolation capabilities would imply that the generated shapes are smooth and plausible variations of the training shapes. We introduce measures to quantitatively evaluate these properties as follows. Note that, these properties are not explicitly optimized for in the loss functions of most neural networks and thus in principle they should not be expected. However, in practice, latent spaces learned by deep networks tend to reflect some of the shape semantics, and thus our goal in this work is to quantify how well these properties are learned.

Semantic organization. In our work, we define the semantic similarity of shapes based on their parameter vectors (defined in Section 4). Since each parameter defines the existence or type of a semantic feature of the shapes, we define the semantic similarity of two shapes as the Euclidean distance between their parameter vectors. For example, two shapes that differ by exactly one binary parameter would have a distance of 1; shapes differing by two binary parameters a distance of $\sqrt{2}$, and so on. Thus, to evaluate the quality of the organization of shapes in the latent space, we compare the Euclidean distances of pairs of shapes in the parameter space to the distance of the corresponding shapes in the latent spaces. Since the two spaces can have different scales and numbers of dimensions, we perform the comparison via the distance correlation [9].

Specifically, given as input two sets of vectors X and Y, where X are the parameter vectors and Y the learned latent vectors for a set of shapes, with $|X| = |Y| = n$, our goal is to compute the correlation among all the pairwise distances $a_{j,k} = \|X_j - X_k\|$ and $b_{j,k} = \|Y_j - Y_k\|$. We can compute this correlation using the distance correlation $\mathrm{dCor}(X, Y)$ [9] defined as:

$$\mathrm{dCor}(X, Y) = \frac{\mathrm{dCov(X,\ Y)}}{\sqrt{\mathrm{dCov}\,(X, X)\mathrm{dCov}(Y, Y)}}, \qquad (1)$$

where dCov is the distance covariance defined as:

$$\mathrm{dCov}^2(X, Y) = \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n} A_{j,k}\, B_{j,k}, \qquad (2)$$

with $A_{j,k}$ and $B_{j,j}$ being doubly-centered distances, i.e., $A_{j,k} = a_{j,k} - \bar{a}_{j,\cdot} - \bar{a}_{\cdot,k} - \bar{a}_{\cdot,\cdot}$, where $\bar{a}_{j,\cdot}$ is the mean of the j-th row, $\bar{a}_{\cdot,k}$ is the mean of the k-th column, and $\bar{a}_{\cdot,\cdot}$ is the mean of all the distances. $B_{j,k}$ is defined in an analogous way for the set Y.

We compute multiple distance correlations per model and dataset by keeping only the N smallest pairwise Euclidean distances in parameter space, and using the same pairwise indices to also filter the corresponding distances in the latent space. We can then compute the distance correlation as described above on the filtered distances. The rationale for this method is that usually the accuracy of distances among shapes that are highly dissimilar is less important than that of similar shapes, which should be grouped closely in the latent space.

Latent space directions. One common manner of evaluating whether the directions in a latent space capture the variation of specific semantic attributes is to vary a single dimension of a latent space and analyze how the attributes change. This type of evaluation has been typically performed only in a qualitative manner, e.g., see Figure 5 of Wu et al. [22]. Given that we have a controlled experimental setting, we generalize this approach to perform a quantitative evaluation of this property.

Specifically, for each dimension d of the latent space, we sort all the parameter vectors for the training shapes according to the values in dimension d of the corresponding latent vectors. Then, we record the number of times different attributes of the parameter space change along this order for all dimensions. If a dimension d groups an attribute in a meaningful manner, then we can expect that the number of changes of the corresponding parameter will be small compared to other parameters along this dimension. Thus, in our evaluation, we count the dimensions where there are parameter changes that are low outlier values, which signify large distributions over the dimension.

To find the low outliers, we identify how many parameters have changing values below the lower quartile, where the lower quartile is simply the middle number between the minimum and median of the change values, as commonly defined. To further filter these dimensions, we only include ones where the number of values greater than the upper

Dataset	3D-GAE	3D-PGAE	3D-AE	3D-VAE
Beds	0.8652	**0.8916**	0.8878	0.8219
Chairs	0.7234	**0.7394**	0.7100	0.6243
Tables	0.6410	**0.6537**	0.6204	0.5641
Shelves	0.7870	**0.8163**	0.8147	0.7378

Table 2: Distance correlation between the parameter vectors and the latent spaces learned with different models. Largest (best) values are marked in bold.

quartile does not exceed the lower quartile count. The existence of low outliers implies that the dimension d groups specific parameter values in a meaningful manner, compared to other dimensions.

Since our parameter vectors include both scalar and binary elements, this creates an imbalance when computing the number of changes, as scalars containing more than two steps will outweigh their binary equivalents. In order to mitigate this, we divide each scalar element by its number of steps, and divide binary elements by 2, so that both types of elements have the same weight on the computation.

Interpolation quality. To evaluate the quality of interpolated shapes, we create new latent vectors according to vector arithmetic, and evaluate how similar the shapes generated from the new latent vectors are to the expected shapes.

We consider only scalar parameters for this evaluation. For example, suppose that we are given the latent vectors for chairs differing only by their leg length. Moreover, suppose that we compute the average of the latent vectors for shapes with leg lengths 0.2 and 0.6. We would expect that the shape generated from this average vector would correspond to a shape with leg length 0.4. Since we have such a shape with leg length 0.4 in the training set, we can compute the difference between the interpolated and ground-truth shape to quantify the quality of the interpolation.

Given that our shapes are encoded as volumes, we simply take the Euclidean distance between these two volumes as a measure of their dissimilarity. Suppose that a shape Z has semantic neighbors X and Y. Following the example above, if Z corresponds to the shape with leg length 0.4, X would correspond to the shape with leg length 0.2 and Y to the shape with leg length 0.6. Then, the distance between the interpolated shape and the expected shape Z is given by $d\left(\frac{X+Y}{2}, Z\right)$ where d is the Euclidean distance between two volumes. In our evaluation, we perform this computation for all the possible intermediate values of all scalar attributes in the dataset (back width, back height, etc., for chairs), and report the average of all experiments. Specifically, this is given by:

$$\text{dInterp} = \frac{1}{n}\sum_{i=1}^{n} d\left(\frac{X_i + Y_i}{2}, Z_i\right), \tag{3}$$

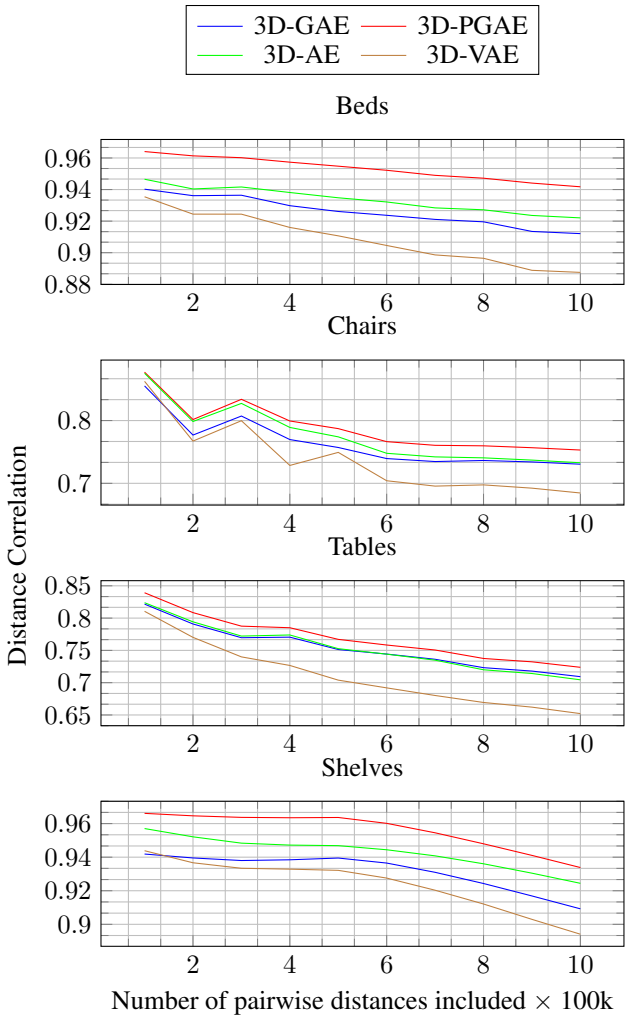


Figure 5: The distance correlation based on the number N of lowest Euclidean distances included in the computation.

where n is the total number of interpolation instances considered (tuples formed by an X_i, Y_i and Z_i). To provide a reference for the values of dInterp, we also report the average of the distances of X_i to Y_i, computed as $d(X_i, Y_i)$.

7. Results

Before training, we first convert the meshes in the dataset into voxelized data of 32^3 voxels. Each pair of a model and dataset was trained for 200 epochs with batch sizes of 10. Most of the models use a learning rate of 0.001, whereas the VAE uses an initial learning rate of 0.0001 with a momentum of 0.9. All networks are trained on a Linux server with an RTX 2080 Ti graphics card equipped with 10 GB of GPU memory and 22 GB of CPU memory.

Semantic organization. Table 2 reports the distance correlation between the parameter vectors and the latent spaces

Dataset	3D-GAE	3D-PGAE	3D-AE	3D-VAE
Beds	88	93	92	**95**
Chairs	17	17	**28**	0
Tables	17	**25**	19	14
Shelves	24	**27**	22	22

Table 3: The number of latent dimensions that group the shape attributes in a meaningful manner, according to an analysis of outliers of parameter changes. Largest (best) values are marked in bold.

learned by different models for different datasets. We see that the learned latent spaces of each model encode the semantic characteristics of the shapes reasonably well for some sets such as *beds*, with correlations over 0.82, while for other sets such as *tables*, the correlations can be as low as 0.56. We also note that the first three AE-based models do not have much of a difference among their correlations, performing much better than the VAE, with the 3D-PGAE model performing the best.

When analyzing the correlations in more detail by selecting only the N smallest Euclidean distances, as seen in Figure 5, we see that the 3D-VAE model produces the lowest correlations for different values of N. We also note that the 3D-PGAE model produces consistently the best results for different values of N, surpassing the 3D-GAE. Finally, certain datasets such as *beds* and *shelves* have large correlations, given their lower diversity in structures.

Latent space directions. In our second experiment, we evaluate the number of dimensions out of 128 that group parameters in a meaningful manner, according to the analysis of low outliers described in Section 6. The results are shown in Table 3. Our results indicate that the quality of the latent space directions depends on the specific category of shapes, as seen with the *beds* dataset, which contains the most latent dimensions where parameters change a small number of times. We also note that, overall, the 3D-PGAE results contain the most amount of meaningful dimensions, which correlates with the semantic organization experiment discussed above.

Interpolation quality. Finally, we show the quality of interpolations between two latent vectors compared to the expected shapes, by calculating the Euclidean distance between the volumes of the interpolated and expected shapes. Table 4 presents the average of all distances for each dataset and architecture, compared to the average distances between the neighboring shapes. We note that, for all the datasets and architectures, the distances between the interpolated vectors and the expected vectors all lie within the distance between their two neighbours, with the *bed* dataset having much lower interpolated distances due to the lower diversity between shapes. In general, the performance of the 3D-GAE is higher or close to that of other models given that the GAE considers neighboring shapes during learning.

In Table 5, we also show selected visual examples of the interpolated reconstructions compared to the expected shapes, where the interpolation is based on the width parameter of the shapes. We do not present examples for the *beds* and *shelves* datasets, as these have limited diversity compared to *tables* and *chairs*. We note that the interpolated shapes tend to be noisy, with the results for the 3D-VAE being of the lowest quality.

Code and datasets. The datasets, as well as the code to compute the evaluation measures, are available at our github page (https://github.com/SharjeelAliCS/3D-latent-space-eval). This page also includes links to the implementations of the different architectures.

8. Discussion

We introduced a dataset of shapes with known semantic structure to better study the quality of the latent spaces learned by various autoencoder models. By studying the latent space organization, we determined that, in general, autoencoder-based networks organize their latent spaces generally to the same high degree of correlation to the semantic parts of the shapes, with no single network performing the best in all experiments. Moreover, we also determined that the VAE provides the lowest-quality latent space organization compared to the other types of autoencoders.

We also evaluated latent space directions to better understand the latent space organization. We demonstrated that each dataset has a specific number of dimensions that organize parameters in a meaningful manner, from which we conclude that the latent spaces are organized fairly well despite some possible redundancy in the dimensions. The latent space organization also varies for types of shapes, such as *beds* having the largest number of meaningful dimensions. This is attributed to the low diversity of *beds* in the parameter vectors, specifically, the low number of $binary$ parameters, which results in each parameter element having a greater impact on the shape.

Our second goal was evaluating the quality of interpolated shapes. We used the scalar attributes of the shapes to create a control group of shapes, which we used to compare interpolated shapes to the expected shapes. We found through both quantitative and qualitative evaluations that the GAE based models provide the most plausible interpolations, while the VAE provides the interpolations of lowest quality, similarly as found in the image synthesis literature.

Finally, we demonstrated that parameter attributes can be

Dataset	3D-GAE		3D-PGAE		3D-AE		3D-VAE	
	Interpolated	Neighbour	Interpolated	Neighbour	Interpolated	Neighbour	Interpolated	Neighbour
Beds	**3.780**	17.126	4.236	18.061	4.205	18.130	10.560	18.924
Chairs	15.668	27.950	15.690	27.955	**15.117**	28.000	19.466	28.123
Tables	**16.666**	24.855	17.238	24.731	16.802	25.340	21.669	26.812
Shelves	21.605	27.216	**21.283**	27.667	22.055	27.684	22.247	27.679

Table 4: Average Euclidean distances between interpolated and expected shapes for different architectures, shown in the columns denoted as "Interpolated", with the lowest (best) values marked in bold. The average distances between the pairs of shapes used in the interpolation are also shown in the columns denoted "Neighbour". In order to be meaningful, the distances between the interpolated and expected shapes should have a smaller value than the distances between the neighbours.

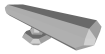
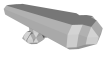
Type	Chairs				Tables			
	3D-GAE	3D-PGAE	3D-AE	3D-VAE	3D-GAE	3D-PGAE	3D-AE	3D-VAE
Neighbour 1								
Interpolated								
Expected								
Neighbour 2								

Table 5: Two examples of interpolations based on a shape's width scalar parameter, compared to the expected shape.

used to constrain the learning of a GAE to provide a better organization of shapes in the latent spaces.

Limitations. One limitation of our evaluation is that the scale of the experiments had to be limited by memory constraints. We had to limit each dataset to 2,200 shapes, and represent the shapes as volumes of at most 32^3 voxels. This resulted in a loss of detail in some of our models, such as the headrests and boards for both *beds* and *chairs*, where fine details such as vertical or horizontal bars could not be represented in the volumes. Thus, we removed these shapes from our training sets. This problem also happens for a few instances of the *shelves* dataset, in which certain shelves become filled instead of being hollow. The model diversity is also low for certain datasets, such as *beds* and shelves, which could be improved upon by adding a greater variety of semantic parts and synthesizing asymmetric shapes.

Another limitation of our evaluation is that we experimented only with volumetric representations of shapes. It would be interesting to also evaluate the encoding of structured representations such as graphs or hierarchies of parts. On the other hand, several recent works still use image and volume-based encoders in their architectures [6], although the decoders can use other types of representations such as implicit functions to generate higher-quality shapes. Thus, our conclusions on the organization of latent spaces would still apply to these hybrid architectures.

Future work. One of our goals for future work is to expand the number of categories of generated datasets by creating procedural rules for other classes of shapes. Another direction for future work is to use higher-resolution volumes of 64^3 voxels to represent shapes, in order to preserve the finer details of the shapes, and possibly use a decoder based on an implicit representation to better assess the quality of shape interpolations.

Acknowledgements. We thank the anonymous reviewers for their comments. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) via an Undergraduate Research Student Award (URSA) and a Discovery Grant.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3D point clouds. In *Proc. Int. Conf. on Machine Learning*, pages 40–49, 2018. 1, 2

[2] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *CoRR*, abs/1608.04236, 2016. 4

[3] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-VAE. In *NIPS Workshop on Learning Disentangled Representations*, 2017. 3

[4] Siddhartha Chaudhuri, Daniel Ritchie, Jiajun Wu, Kai Xu, and Hao Zhang. Learning generative models of 3D structures. *Computer Graphics Forum*, 39(2):643–666, 2020. 1, 2

[5] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. BSP-NET: Generating compact meshes via binary space partitioning. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2020. 1, 3

[6] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 1, 2, 8

[7] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[8] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *Proc. Int. Conf. on Learning Representations*, 2017. 1, 3

[9] Dominic Edelmann, Tamás F. Móri, and Gábor J. Székely. On relationships between the pearson and the distance correlation coefficients. *Statistics & Probability Letters*, 169:108960, 2021. 5

[10] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3D object reconstruction from a single image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2463–2471, 2017. 4

[11] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. A papier-mâché approach to learning 3D surface generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 216–224, 2018. 1, 2

[12] Yanran Guan, Tansin Jahan, and Oliver van Kaick. Generalized autoencoder for volumetric shape generation. In *CVPR Workshop on Learning 3D Generative Models*, pages 1082–1088, 2020. 4

[13] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 3

[14] R. Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy Mitra, and Daniel Ritchie. ShapeAssembly: learning to generate programs for 3D shape structure synthesis. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, 39(6):234:1–20, 2020. 1

[15] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proc. Int. Conf. on Machine Learning*, pages 1558–1566, 2016. 1, 3

[16] Paul Merrell and Dinesh Manocha. Model synthesis: A general procedural modeling algorithm. *IEEE Trans. on Visualization and Computer Graphics*, 17(6):715–728, 2011. 3

[17] C. Nash and C. K. I. Williams. The shape variational autoencoder: A deep generative model of part-segmented 3D objects. *Computer Graphics Forum*, 36(5):1–12, 2017. 1

[18] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 1, 2

[19] Ruben M. Smelik, Tim Tutenel, Rafael Bidarra, and Bedrich Benes. A survey on procedural modelling for virtual worlds. *Computer Graphics Forum*, 33(6):31–50, 2014. 1, 3

[20] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *Proc. Int. Conf. on Learning Representations*, 2016. 3

[21] Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. Instant architecture. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 22(3):669–677, 2003. 3

[22] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pages 82–90, 2016. 1, 2, 5