Learning to Generate the “Unseen” via Part Synthesis and Composition

Nadav Schor  
Tel Aviv University  
Israel  
nschor@gmail.com

Oren Katzir  
Tel Aviv University  
Israel  
okaztir@gmail.com

Hao Zhang  
Simon Fraser University  
Canada  
haoz@cs.sfu.ca

Daniel Cohen-Or  
Tel Aviv University  
Israel  
dcor@tau.ac.il

“For the task of object recognition, the visual system decomposes shapes into parts, ..., and parts with their descriptions and spatial relations provide a first index into a memory of shapes.” — Hoffman & Richards [16]

Abstract

Data-driven generative modeling has made remarkable progress by leveraging the power of deep neural networks. A reoccurring challenge is how to sample a rich variety of data from the entire target distribution, rather than only from the distribution of the training data. In other words, we would like the generative model to go beyond the observed training samples and learn to also generate “unseen” data. In our work, we present a generative neural network for shapes that is based on a part-based prior, where the key idea is for the network to synthesize shapes by varying both the shape parts and their compositions. Treating a shape not as an unstructured whole, but as a (re-)composable set of deformable parts, adds a combinatorial dimension to the generative process to enrich the diversity of the output, encouraging the generator to venture more into the “unseen”. We show that our part-based model generates richer variety of feasible shapes compared with a baseline generative model. To this end, we introduce two quantitative metrics to evaluate the ingenuity of the generative model and assess how well generated data covers both the training data and unseen data from the same target distribution.

1. Introduction

Learning generative models of shapes and images has been a long standing research problem in visual computing. Despite the remarkable progress made, an inherent and reoccurring limitation still remains: a generative model is often only as good as the given training data, as it is always trapped or bounded by the empirical distribution of the observed data. Yet, the generative power of a learned model should not only be judged by the plausibility of the data it can generate, but also its diversity, in particular, by the model’s ability to generate data from the same underlying distribution but that is sufficiently far removed from the training set. The main challenge is how to develop a generative model that effectively learns to generate the “unseen”. Even statistical evaluation of such a model is a non-trivial task since the target distribution that encompasses the observed and unseen data is unknown.

We believe that the key to generative diversity is to enable more drastic changes, i.e., non-local and/or structural transformations, to the training data. At the same time, such changes must be within the confines of the target data distribution. In our work, we focus on generative modeling of 2D or 3D shapes, where the typical modeling constraint is to produce shapes belonging to the same category as the exemplars. To this end, we develop a generative deep neural

Figure 1: Our part-based generative model (b) covers significantly more the “unseen” data than the baseline (a). Generated data (pink dots) by two methods are displayed over training data (purple crosses) and unseen data (green crosses) from the same target distribution. The data is displayed via PCA over a classifier feature space, with the three distributions summarized by ellipses for illustration only. A few samples of training, unseen, and generated data are displayed to reveal their similarity/dissimilarity.
network based on a part-based prior. That is, we assume that shapes in the target distribution are composed of parts, e.g., chair backs or airplane wings. The network is designed to synthesize novel parts, independently, and then compose them to form a complete coherent shape.

It is well-known that object recognition is intricately tied to reasoning about parts and part relations [16, 36]. Hence, building a generative model based on varying parts and their compositions, while respecting category-specific part priors, is a natural choice and also facilitates grounding of the generated data to the target object category. More importantly, treating a shape as a (re-)composable set of parts, instead of a whole entity, adds a combinatorial dimension to the generative model and improves its diversity. By synthesizing parts independently and then composing them, our network enables both part variation and novel combination of parts, which induces non-local and more drastic shape transformations. Rather than sampling only a single distribution to generate a whole shape, our generative model samples both the geometric distributions of individual parts and the combinatorial varieties arising from part compositions, which encourages the generative process to venture more into the “unseen”, as shown in Figure 1.

While the part-based approach is generic and not strictly confined to specific generative network architecture, we develop a generative autoencoder to demonstrate its potential. Our generative AE consists of two parts. In the first, we learn a distinct part-level generative model. In the second stage, we concatenate these learned latent representation with a random vector, to generate a new latent representation for the entire shape. These latent representations are fed into a conditional parts compositional network, which is based on a spatial transformer network (STN) [20].

We are not the first to develop deep neural networks for part-based modeling. Some networks learn to compose images [25, 3] or 3D shapes [21, 5, 44], by combining existing parts sampled from a training set or provided as input to the networks. In contrast, our network is fully generative as it learns both novel part synthesis and composition. Wang et al. [37] train a generative adversarial network (GAN) to produce semantically segmented 3D shapes and then refine the part geometries using an autoencoder network. Li et al. [24] train a VAE-GAN to generate structural hierarchies formed by bounding boxes of object parts and then fill in the part geometries using a separate neural network. Both of these works take a coarse-to-fine approach and generate a rough 3D shape holistically from a noise vector. In contrast, our network is trained to perform both part synthesis and part composition (with noise augmentation); see Figure 2.

We show that our part-based model generates a richer variety of feasible shapes compared with a baseline generative model. In addition, to evaluate the generative power of our network relative to baseline approaches, we introduce two quantitative metrics to assess how well the generated data covers both the training data and the unseen data from the same target distribution.

2. Background and Related Work

2.1. Generative neural networks

In recent years, generative neural networks gained a lot of research attention, within the deep learning frameworks. Two of the most commonly used deep generative models are variational auto-encoders (VAE) [23] and generative adversarial networks (GAN) [13]. Both methods have made remarkable progress in image and shape generation problems [39, 19, 32, 45, 38, 41, 37].
Figure 3: Novel shape generation at inference time. We randomly sample the latent spaces for shape parts and part compositions. Using the pre-trained part-decoders and the composition network, we generate novel parts and then warped them to produce a coherent whole shape.

Many researches have delved into these generative processes, improving training and extending the basic models. In [14, 27, 4], new cost functions are suggested to achieve smooth and non-vanishing gradients. Sohn et al. [34] and Odena et al. [30] proposed conditional generative models, based on VAE and GAN, respectively. Hoang et al. [15] train multiple generators to explore different modes of the data distribution. Similarly, MIX+GAN [2] uses a mixture of generators to improve diversity of the generated distribution, while a combination of multiple discriminators and a single generator aims at constructing a stronger discriminator to guide the generator. GMAN proposed in [8] explores an array of discriminators to boost the learning of a generator. Some methods [18, 26, 43] use a global discriminator together with multiple local discriminators.

Generative model was also proposed directly on point clouds. Following the introduction of PointNet [31], a neural network for processing point clouds, Achlioptas et al. [1] proposed an AE+GMM generative model for point clouds, which is considered state-of-the-art.

Our work is orthogonal to these methods. We address the case where the generator is unable to generate other valid samples since they are not represented in the training data. We show that by using prior information we can assist the generation process and extend the generator capabilities.

2.2. Learning-based shape synthesis

Li et al. [24] present a top-down approach, focusing more on the part structure for 3D shape generation. They learn symmetry hierarchies of shapes with an autoencoder and then generate variations of these hierarchies, using an adversarial discriminator. The nodes of the hierarchies are independently instantiated with parts. However, these parts are not necessarily connected and their aggregation does not form a coherent connected shape. In our work, the shapes are generated coherently as a whole, and special care is given to the inter-parts relation and their connectivity.

Most relevant to our work is the shape variational auto-encoder by Nash and Williams [29]. They developed an auto-encoder to learn a low-dimensional latent space, so that novel shapes can be generated by sampling vectors in the learned space. Like our method, the generated shapes are segmented into the relevant semantic parts. In contrast however, they require a one-to-one dense correspondence among the training shapes, since they represent the shapes as an order vector. Their auto-encoder learns the overall (global) 3D shapes with no attention to the local details. Our approach pays particular attention to both the generated surface and the relation in-between local shape parts.

2.3. Assembly-based shape synthesis

There are numerous works that create new models by assembling from existing components. The pioneer work of Funkhouser et al. [12] composes shapes by retrieving relevant shapes from a repository, then cuts and extracts components from these shapes and glues them together to form a new shape. The following works [35, 6, 21, 40, 22, 10, 17] try to improve the modeling process with more sophisticated techniques that consider the part relations or shape structures, e.g., employing Bayesian networks or modular templates. We refer to a STAR report [28] for an overview of works on this aspect. Recent works [25, 3] suggested the use of neural-networks to assemble images or scenes from existing components. These works utilized a Spatial Transformation Networks (STNs) [20] to compose the existing components to a coherent image/scene. In our work, STN is integrated as an example for prior information regarding the data generation process. In contrary to previous works, we first synthesize parts using multiple generative AEs and then we use STN to compose the parts.

3. Method

In this section, we present our generative model which learns to synthesize shapes that can be represented as a composition of distinct parts. We assume that the parts are independent of each other. Thus, every combination of parts is valid, even if the training set may not include it. As shown in Figure 2, the generative model consists of two units. The first is a generative model of parts, and the second, combines the generated parts into a global shape.

3.1. Part synthesis

We first train a generative model that estimate the marginal distribution of each part separately. In the 2D case, we use a standard VAE as the part generative model, and train different VAEs for each individual semantic part. Thus, each part is fed into a different VAE and is mapped onto a separate latent distribution. The encoder consists of
several convolution layers followed by Leaky-ReLU activation functions. The final layer of the encoder is a fully connected layer producing the latent distribution parameters. Using the reparameterization trick, the latent distribution is sampled and decoded to reconstruct each individual input part. The decoder mirrors the encoder network, applying a fully connected layer followed by transposed convolution layers with ReLU non-linearity functions. In the 3D case, we borrow an idea from Achlioptas et al. [1], and replace the VAE with AE+GMM (we approximate the latent space of the AE by using GMM). The encoder is based on PointNet [31] and the decoder consists of fully-connected layers. The part synthesis process is visualized in Figure 2, part synthesis unit.

Once the part synthesis unit is trained, the part encoders are fixed, and used to train the part composition unit.

### 3.2. Parts composition

This unit composes the different parts into a coherent shape. Given a shape and its parts, the pre-trained encoders generate codes for each part (marked in blue in Figure 2). At training time, these generated codes are fed into a composition network which learns to produce a transformation per part (scale and translation) such that the composition of all the parts forms a coherent complete shape. The loss measures the similarity between the input shape and the composed shape. We use Intersection-over-Union (IoU) as our metric in the 2D domain, and Chamfer distance ($d_C$) for 3D shapes, which is given by

$$d_C(Q, P) = \sum_{q \in Q} \min_{p \in P} (q - p)^2 + \sum_{p \in P} \min_{q \in Q} (p - q)^2. \quad (1)$$

Note that the composition network yields a set of affine (similarity) transformations which are applied on the input parts, and does not directly synthesize the output shape.

The composition network does not learn the composition based solely on part codes, but also relies on an input noise vector. This network is another generative model on its own, generating the scale and translation from the noise, conditioned on the code of the parts. This additional generative model enriches the variation of the generated model, beyond the generation of the parts.

### 3.3. Novel shape generation

At inference time, we sample the composition vector from the normal distribution. In the 2D case, since we use VAE, we sample the parts’ vectors from the normal distribution as well. In the 3D case, we sample the vector of each part from its GMM distribution (randomly picking one Gaussian and sampling from it). From that compound latent vector, we synthesize a shape (see Figure 3). We feed each section of the latent vector, which represents a part, to its associated pre-trained decoder (from the part synthesis unit), and generate novel parts. In parallel, the entire shape representation vector is fed to the composition network to generate a scale and translation for each part. The synthesized parts are then warped according to the generated transformations, and combined to produce a novel shape.

### 4. Architecture and implementation details

The backbone architecture of our part based synthesis is an AE, VAE for 2D and AE+GMM for 3D.

#### 4.1. Part-based generation

**2D Shapes.** Our input parts are assumed to have a size of $64 \times 64 \times 1$. We denote $C(k)$ ($TC(k)$) as a 2D convolution (transpose convolution) layer with $k$ filters of size $5 \times 5$ and stride 2, followed by batch normalization and a leaky-ReLU (ReLU). We denote $L(k)$ to be a fully-connected layer with $k$ output nodes. The encoder takes a part image with 1 channel as input. The encoder structure is $C(8) - C(16) - C(32) - C(64) - L(10)$. The decoder mirrors the encoder $L(1024) - TC(32) - TC(16) - TC(8) - TCS(1)$, where in the last layer $TCS$ we omitted the batch normalization layer and replaced the ReLU activation function with a sigmoid. The output of the decoder is equal in size to the input ($64 \times 64 \times 1$). We use an Adam optimizer with learning rate $= 2e^{-4}$, $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The batch size is set to 64.

**3D point clouds.** Our input parts are assumed to have a fixed number of point for each part, we used 400 points per part. We denote MP as a feature-wise max-pooling layer and $1DC(k)$ as a 1D convolution layer with $k$ filters of size 1 and stride 1, followed by a batch normalization layer and a ReLU. The encoder takes a part with $400 \times 3$ as input. The encoder structure is $1DC(64) - 1DC(64) - 1DC(128) - 1DC(64) - MP$. The decoder consists of fully-connected layers. We denote $L(k)$ to be a fully-connected layer with $k$ output nodes, followed by a batch normalization layer and a ReLU. The decoder takes a latent vector of size 64 as input. The decoder structure is $L(256) - L(256) - LC(400 \times 3)$, where in the last layer $LC$ we omitted the batch-normalization layer and the ReLU activation function. The output of the decode is equal in size to the input ($400 \times 3$). We use a GMM, with 20 Gaussians, for each AE, to model their latent space distribution. We use Adam optimizer with learning rate $= 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The batch size is set to 64.

#### 4.2. Part composition

**2D.** The composition network encodes each part by the trained VAE encoder, producing 10-dim vector for each part. The composition noise vector is set to size 8. The parts encoded vectors are concatenated together with the noise vector, yielding a 48-dim vector. The composition network
structure is $L(128) - L(128) - L(16)$. Each fully connected layer is followed by a batch normalization layer, ReLU activation function and a Dropout layer with keep rate of 0.8, except from the last layer. The last layer outputs a 16-dim vector, four values per part. These four values represent the scale and translation in $x$ and $y$ axes. We use the grid generator and sampler, suggested by [32], for performing the differential transformation process. The scale is initialized to 1 and the translation to 0. We use per-part IoU loss, optimize by Adam optimizer with learning rate $= 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The batch size is set 64.

3D. The composition network encodes each part by the trained AE encoder, producing 64-dim vector for each part. The composition noise vector is set to size 16. The parts encoded vectors are concatenated together with the noise vector, yielding a 272-dim vector. The composition network structure is $L(256) - L(128) - L(24)$. Each fully connected layer is followed by a batch normalization layer, and a ReLU activation function, except from the last layer. The last layer outputs a 24-dim vector, six values per part. These six values represent the scale and translation in $x$, $y$ and $z$ axes (we initial the scale 1 and the translation to 0). We then reshape the output vector to match an affine transformation matrix:

$$
M = \begin{bmatrix}
  s_x & 0 & 0 & t_x \\
  0 & s_y & 0 & t_y \\
  0 & 0 & s_z & t_z
\end{bmatrix}
$$

(2)

The task of performing an affine transformation on point clouds is easy, we simply concatenate 1 to each point $(x, y, z, 1)$ and multiply the transformation matrix with each point. We use Chamfer distance loss and, optimize by Adam optimizer with learning rate $= 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The batch size is set 64.

5. Results and evaluation

In this section, we analyze the results of applying our generative approach on images and 3D shape collections.

5.1. Datasets

Projected COSEG. We used the COSEG dataset [33] which consists of 300 vases, segmented to four different semantic labels: top, handle, body and base (each vase may or may not contain any of these parts). Similar to the projection procedure in [11], each vase is projected from the main view to constitute a collection of 300 silhouettes of size $64 \times 64$, where each semantic part is stored in a different channel. In addition, we create four sets, one per part. The parts are normalized by finding their axis-aligned bounding box and stretching it to a $64 \times 64$ resolution.

Shape-Net. For 3D data, we chose to demonstrate our method on point clouds taken from ShapeNet part dataset [42]. We chose to focus on two categories: chairs and airplanes. Point clouds, compared to 3D voxels, enable higher resolution while keeping the model complexity relatively low. Similar to the 2D case, each shape is divided into its semantic parts (Chair: legs, back, seat and arm-rests, Airplane: tail, body, engine and wings). We first normalize each shape to the unit square. We require an equal number of points $N$ in each point cloud, thus, we randomly sample each part to $N = 400$ points. If a part consists of $M < N$ points, we randomly duplicate $N - M$ of its points (since our non-local operation performs only max global pooling, the duplication of points has no affect on the embedding of the shape). This random sampling process occurs every epoch. For consistency between the shape and its parts, we first normalize the original parts to the unit square, and only then sample (or duplicate) the same points that were selected to generate the complete sampled shape.

We divide the resulting collections into two subsets: (i) training set and (ii) unseen set. The term unseen emphasizes that unlike the nominal division into train and test sets, the unseen set is not represented well in the training set. Thus, the unseen set is used to evaluate the ability of the model to generate new and diverse shapes.

5.2. Baselines

For 2D, we use a naive model - a one-channel VAE. Its structure is identical to the parts VAE with a latent space of 48-dim. We feed a binary representation of the data (a silhouette) as input. We use an Adam optimizer with learning rate $= 2e^{-4}$, $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The batch size is set to 64.

In the 3D case, we use [1] model, which generates remarkable 3D point cloud results. We train their model using our 400-points per part data set ($1600 \times 3$ per shape). We use their official implementation and parameters.

5.3. Qualitative evaluation

We evaluate our method on 2D data and 3D points clouds. In Fig. 4 some of our generated results for 3D points clouds are presented. Unlike other naive generative approaches, we are able to generate versatile samples beyond the empirical distribution. In order to visualize the versatility, we present the nearest neighbor of the generated samples in the training set. As shown in Fig. 5, for the 2D case, sampled generated by our generative approach differ from the closest training samples. In Fig. 6 we also compare this qualitative diversity measure with the baseline, showing that our generated samples are more distinct from their nearest neighbors in the training set, compared to those generated by the baseline. In the following section we quantify

\[1\]https://github.com/optas/latent_3d_points
Figure 4: Representative samples of our generated results in 3D. Our part-base generation and assembly process produces realistic and diverse results.

Figure 5: A gallery of 10 randomly sampled vases generated by our method (top row) and their 3-nearest-neighbors (below) from the training set, based on pixel-wise Euclidean distance. One can observe that the generated vases are different from their neighbors.

5.4. Quantitative Evaluation

We quantify the ability of our model to generate realistic unseen samples using two novel metrics.

**k-set-coverage.** We define the $k$-set-coverage of set $A$ by set $B$ as the percentage $P$ of shapes from $A$ which are one of the $k$-nearest-neighbors of some shape in $B$. Thus, if the set $B$ is similar only to a small part of set $A$, the $k$-Set-Coverage will be small and vice-versa. In our case, we calculate the nearest neighbors using Chamfer distance. In Fig. 7, we compare the $k$-set-coverage of the unseen set and the training set by our generated data and by the baseline generated data. It is clear that the baseline covers the training better, since most of its samples lie close to it. However, the unseen set is covered poorly by the baseline, for all $k$, compared to our part-based generative approach.

**Diversity.** We develop a second measure to quantify the generated unseen data, which relies on a trained classifier to distinguish between the training set and the unseen set. Then, we measure the percentage of generated shapes which are classified as belonging to the unseen set. The classifier architecture is a straightforward adaptation of the encoder from the part synthesis unit of the training process, followed by fully connected layers which classify between unseen and train sets (see supplementary file for more details). Tab. 1 summarizes the classification for different datasets. From Tab. 1 we learn that our generative samples are classified to the unseen set more than the baseline generated shapes. To visualize these results, we use the classifier’s embedding (the layer before the final fully-connected layer) and reduce its dimension by projecting it onto the 2D PCA plane, as shown in Figs. 1 and 8. The training and unseen sets have overlap in this representation, reflecting data which is similar between the two sets. While both methods are able to generate unseen samples in the overlap region, the baseline samples are biased toward the training set. In contrary, our generated samples are closer to the unseen.

| Dataset          | Train   | Unseen   |
|------------------|---------|----------|
|                  | VAE     | Ours     | [1] model | Ours |
| COSEG            | 0.82    | 0.5      | 0.18      | 0.5  |
| Shape-Net chairs | 0.44    | 0.24     | 0.56      | 0.76 |
| Shape-Net airplanes | 0.43   | 0.14     | 0.57      | 0.86 |

Table 1: Classification results of generated data between train and unseen sets.

5.5. Latent space interpolation

Our model consists of independent generative processes, namely each individual part and the composition. Each generative model has its own individual latent space. Thus, our model enables additional control over each individual gen-
Figure 6: Qualitative comparison of diversity. The generated data of both methods (first row) is realistic. However, searching for the nearest neighbors of the generated data in the training set (rows two to four) reveals that our method (right side) exhibits more diversity compared to the baseline (left side). Please note that the baseline’s shapes are presented in gray to emphasize that the baseline generates the entire shape.

Figure 7: $k$-set-overage comparison for chairs (left) and airplanes (right) point clouds sets. We generate an equal number of samples in both our method and the baseline [1] and calculate the $k$-set-coverage of both the training set and unseen set by them. While the baseline covers the training set almost perfectly, it has lower coverage of the unseen set.

Figure 8: Classifier’s feature space for 3D data. The classifier is trained to distinguish between training set (purple) and the unseen set (green). The two sets are clearly visible in the resulting space. While the baseline method [1] generates samples similar to the seen set, our method generates samples in the unseen region.

6. Conclusion, limitation, and future work

We believe that effective generative models should strive to venture more into the “unseen” data of a target distribution, beyond the observed exemplars from the training set.
Figure 9: Complete latent space interpolation (parts and composition). The shape’s code on the left is linearly interpolated to the shape’s code on the right. The complete latent space between two shapes (parts and composition).

Figure 10: Interpolation by parts. Our generation process enables discrete interpolation between two shapes, one part at a time. The changed part from left to right: back, seat, legs and armrests.

Figure 11: Composition code manipulation. By changing only the composition code, we generate a different yet realistic shape using the same parts. Rows one to four present the top, handle, body and base (in this order). The last row presents the complete vase.

Covering both the seen and the unseen implies that the generated data is both fit and diverse [40]. Fitness constrains the generated data to be close to data from the target domain, both the seen and the unseen. Diversity ensures that the generated data is not confined only to the seen data.

We have presented a generic approach for “fit and diverse” shape modeling based on a part-based prior, where a shape is not viewed as an unstructured whole but as the result of a coherent composition of a set of parts. This is realized by a novel deep generative network composed of a part synthesis unit and a part composition unit. Novel shapes are generated via inference over random samples taken from the latent spaces of shape parts and part compositions. Our work also contributes two novel measures to evaluate generative models: the $k$-set-coverage and a diversity measure which quantifies the percentage of generated data classified as “unseen” vs. data from the training set.

Compared to a baseline approach, our part-based generative network demonstrates superior, but still somewhat limited, diversity, since the generative power of the part-based approach is far from being fully realized. Foremost, our composition mechanism is still very much “in place” as it does not allow changes to part structures or feature transfers between different part classes. For example, enabling a simple symmetric switch in the part composition would allow our network to generate right hand images when all the training images are of the left hand.

Our current method is also limited by the spatial transformations allowed by the STN we employ during part composition. In addition, our current network is still designed based on a similarity loss with respect to the training shapes, which limits diversity. One possible remedy is to allow the validation set to be much richer than the training set for part generation and composition. For example, speaking in the language of GANs, we could define a generator for 3D shapes, but a discriminator utilizing images.

As more immediate future work, we would like to apply our approach to more complex datasets, where parts can be defined during learning. In general, we believe that more research shall focus on other generation related prior information, besides parts-based priors, and means to incorporate them into the generation process. Further down the line, we envision that the fit-n-diverse approach will form a baseline for creative modeling [7]. The compelling challenge is how to define a generative neural network with sufficient diver-
References

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3D point clouds. 2018. 3

[2] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. Generalization and equilibrium in generative adversarial nets (GANs). In Proc. Int. Conf. on Machine Learning, volume 70, pages 224–232. 2017. 3

[3] S. Azadi, D. Pathak, S. Ebrahimi, and T. Darrell. Compositional gan: Learning conditional image composition. arXiv preprint arXiv:1807.07560, 2018. 2

[4] D. Berthelot, T. Schumm, and L. Metz. Began: boundary equilibrium generative adversarial networks. arXiv preprint arXiv:1703.10717, 2017. 3

[5] S. Chaudhuri, E. Kalogerakis, L. Guibas, and V. Koltun. Probabilistic reasoning for assembly-based 3D modeling. ACM Trans. Gr., 30(4):35:1–35:10, 2011. 2

[6] S. Chaudhuri and V. Koltun. Data-driven suggestions for creativity support in 3D modeling. ACM Trans. Gr., 29(6):183:1–183:10, 2010. 3

[7] D. Cohen-Or and H. Zhang. From inspired modeling to creative modeling. The Visual Computer, 32(1):1–8, 2016. 8

[8] I. Duruˇgkar, I. Gemp, and S. Mahadevan. Generative multi-adversarial networks. arXiv preprint arXiv:1611.01673, 2016. 3

[9] A. M. Elgammal, B. Liu, M. Elhoseiny, and M. Mazzone. CAN: creative adversarial networks, generating ”art” by learning about styles and deviating from style norms. CoRR, abs/1706.07068, 2017. 9

[10] N. Fish, M. Averkiou, O. van Kaick, O. Sorkine-Hornung, D. Cohen-Or, and N. J. Mitra. Meta-representation of shape families. ACM Trans. Gr., 33(4):34:1–34:11, 2014. 3

[11] N. Fish, O. van Kaik, A. Bermano, and D. Cohen-Or. Structure-oriented networks of shape collections. ACM Transactions on Graphics (TOG), 35(6):171, 2016. 5

[12] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefker, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. ACM Trans. Gr., 23(3):652–663, 2004. 3

[13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. pages 2672–2680, 2014. 2

[14] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. pages 5769–5779, 2017. 3

[15] Q. Hoang, T. D. Nguyen, T. Le, and D. Phung. Multi-generator generative adversarial nets. arXiv preprint arXiv:1708.02556, 2017. 3

[16] D. D. Hoffman and W. A. Richards. Parts of recognition. Cognition, pages 65–96, 1984. 1

[17] H. Huang, E. Kalogerakis, and B. Marlin. Analysis and synthesis of 3D shape families via deep-learned generative models of surfaces. 34(5):25–38, 2015. 3

[18] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. ACM Trans. Gr., 36(4):107:1–107:14, 2017. 3

[19] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. pages 5967–5976, 2017. 2

[20] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In Advances in neural information processing systems, pages 2017–2025, 2015. 2

[21] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun. A probabilistic model for component-based shape synthesis. ACM Trans. Gr., 31(4):55:1–55:11, 2012. 2

[22] V. G. Kim, W. Li, N. J. Mitra, S. Chaudhuri, S. DiVerdi, and T. Funkhouser. Learning part-based templates from large collections of 3D shapes. ACM Trans. Gr., 32(4):70:1–70:12, 2013. 3

[23] D. P. Kingma and M. Welling. Auto-encoding variational bayes. 2014. 2

[24] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. Zhang, and L. Guibas. Grass: Generative recursive autoencoders for shape structures. ACM Transactions on Graphics (TOG), 36(4):52, 2017. 2

[25] C. Lin, E. Yumer, O. Wang, E. Shechtman, and S. Lucey. St-gan: Spatial transformer generative adversarial networks for image compositing. In Computer Vision and Pattern Recognition, 2018. CVPR 2018. IEEE Conference on, pages -. -. 2018. 2

[26] G. Liu, F. A. Reda, K. J. Shih, T. -C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. arXiv preprint arXiv:1804.07723, 2018. 3

[27] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. In Computer Vision (ICCV), 2017 IEEE International Conference on, pages 2813–2821. IEEE, 2017. 3

[28] N. Mitra, M. Wand, H. R. Zhang, D. Cohen-Or, V. Kim, and Q.-X. Huang. Structure-aware shape processing. In SIGGRAPH Asia 2013 Courses, pages 1:1–1:20, 2013. 3

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

[30] A. Odena. Semi-supervised learning with generative adversarial networks. arXiv preprint arXiv:1606.01583, 2016. 3

[31] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1609.00672, 2016. 3

[32] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015. 2

[33] O. Sidi, O. van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering, volume 30. ACM, 2011. 5

[34] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In Advances in Neural Information Processing Systems, pages 3483–3491, 2015. 3

[35] J. Talton, L. Yang, R. Kumar, M. Lim, N. Goodman, and R. Méch. Learning design patterns with bayesian grammar
induction. In Proc. ACM Symp. on User Interface Software and Technology, pages 63–74, 2012. 3

[36] D. W. Thompson. On Growth and Form. Dover reprint of 1942 2nd ed., 1992. 2

[37] H. Wang, N. Schor, R. Hu, H. Huang, D. Cohen-Or, and H. Hui. Global-to-local generative model for 3d shapes. ACM Transactions on Graphics (TOG), 37(6):00, 2018. 2

[38] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. pages 318–335, 2016. 2

[39] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. pages 82–90, 2016. 2

[40] K. Xu, H. Zhang, D. Cohen-Or, and B. Chen. Fit and diverse: Set evolution for inspiring 3D shape galleries. ACM Trans. Gr., 31(4):57:1–57:10, 2012. 3, 8

[41] X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2image: Conditional image generation from visual attributes. pages 776–791, 2016. 2

[42] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas. A scalable active framework for region annotation in 3D shape collections. ACM Trans. Gr., 35(6):210:1–210:12, 2016. 5

[43] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. pages 5505–5514, 2018. 3

[44] C. Zhu, K. Xu, S. Chaudhuri, R. Yi, and H. Zhang. SCORES: Shape composition with recursive substructure priors. ACM Transactions on Graphics, 37(6), 2018. 2

[45] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. pages 597–613, 2016. 2
A. Supplementary Material

A.1. Network Architectures

Part synthesis  The architectures of the part synthesis generative autoencoders, for both 3D and 2D cases, are listed in Table 2 and Table 3 respectively. We used the following standard hyper-parameters to train the 3D (2D) model: Adam optimizer, $\beta_1 = 0.9(0.5)$, $\beta_2 = 0.999$, learning rate $= 0.001(2e^{-4})$, batch size $= 64$.

| Operation | Kernel | Strides | Feature maps | Act. func. |
|-----------|--------|---------|--------------|------------|
| Encode $x$ | 400x3 point cloud | $\rightarrow$ | 64-dim feature vector $f_x$ | |
| 1D conv. | 1x64 | 1 | 400x64 | Relu |
| 1D conv. | 1x64 | 1 | 400x64 | Relu |
| 1D conv. | 1x64 | 1 | 400x64 | Relu |
| 1D conv. | 1x128 | 1 | 400x128 | Relu |
| 1D conv. | 1x64 | 1 | 400x64 | Relu |
| Max Pooling | – | – | 128 | – |
| Decode $f_x$ | 64-dim feature vector | $\rightarrow$ | 400x3 point cloud $x'$ | |
| Linear | – | – | 256 | Relu |
| Linear | – | – | 256 | Relu |
| Linear | – | – | 400x3 | – |

Table 2: Layers of the 3D part generative AE shown in order from input to output.

| Operation | Feature maps | Act. func. |
|-----------|--------------|------------|
| Comp. net $x$ | 64xC+16 feature vector | $\rightarrow$ 6XC comp. vector $f_x$ | |
| Linear | 256 | Relu |
| Linear | 128 | Relu |
| Linear | 6xC | – |

Table 4: Layers of the 3D parts composition network shown in order from input to output. Where $C$ is the number of parts. The input vector is a concatenation of $C$ feature vectors of size 64 with a noise vector of size 16.

| Operation | Feature maps | Act. func. |
|-----------|--------------|------------|
| Comp. net $x$ | 10xC+8 feature vector | $\rightarrow$ 4XC comp. vector $f_x$ | |
| Linear | 128 | Relu |
| Linear | 128 | Relu |
| Linear | 4xC | – |

Table 5: Layers of the 2D parts composition network shown in order from input to output. Where $C$ is the number of parts. The input vector is a concatenation of $C$ feature vectors of size 10 with a noise vector of size 8.

A.2. More results

In this section, we present additional results of our model for the three categories. For the Chair and Airplane categories we have randomly sampled 80 models from the 5,000 we have generated for the quantitative metrics; see Fig. 13 and Fig. 14 respectively. For the Vases, since the output is smaller, we sampled 300 models from the 1,024 we generated for the quantitative metrics; see Fig. 15. Furthermore, we present additional interpolation results from our model on the 3D categories; see Fig. 16 and Fig. 18 for linear interpolations, Fig. 17 and Fig. 19 for part-by-part interpolations.

A.3. Comparison to baseline

In this section, we present randomly picked generated results from our model and the baseline on the Chair and Airplane categories. For each generated model, we present its three nearest neighbors based on the Chamfer distance. We present the results side by side to emphasize the power of our model in generating novel shapes in comparison to the baseline; see Fig. 20 for chairs and Fig. 21 for airplanes.

Parts composition  The architectures of the parts composition units are listed in Table 4 and Table 5, for the 3D and 2D cases respectively. We used the following standard hyper-parameters to train the 3D and 2D models: Adam optimizer, $\beta_1 = 0.9$, $\beta_2 = 0.999$, learning rate $= 0.001$, batch size $= 64$. |
Figure 13: Chair gallery. We present 80 randomly sampled chairs from the 5,000 which were generated by our method.
Figure 14: Airplane gallery. We present 80 randomly sampled chairs from the 5,000 which were generated by our method.
Figure 15: Vases gallery. We present 300 randomly sampled vases from the 1,024 which were generated by our method.
Figure 16: Complete latent space interpolation (parts and composition) for the chair category. The shape’s code on the left is linearly interpolated to the shape’s code on the right.
Figure 17: Chairs part-by-part interpolation. In each row we present part-by-part interpolation between two chairs (see left and right objects). The order of replacement is back-seat-legs-armrests. Please note, for certain pairs the arm-rests part does not exist, thus the interpolation of the arm-rest part is identity (no change).
Figure 18: Complete latent space interpolation (parts and composition) for the airplane category. The shape’s code on the left is linearly interpolated to the shape’s code on the right.
Figure 19: Airplanes part-by-part interpolation. In each row we present part-by-part interpolation between two airplanes (see left and right objects). The order of replacement is body-wings-tail.
Figure 20: Chairs nearest-neighbors. We present random generated results from our model and the baseline, for each results, we present its three nearest-neighbors based on the Chamfer distance. In the left half we show results from our model (in colors), where the left column is generated and the next three columns to its right are its three nearest-neighbors. In the right half we present results from the baseline (gray), where the left column is generated and the next three columns to its right are its three nearest-neighbors.
Figure 21: Airplanes nearest-neighbors. We present random generated results from our model and the baseline, for each results, we present its three nearest-neighbors based on the Chamfer distance. In the left half we show results from our model (in colors), where the left column is generated and the next three columns to its right are its three nearest-neighbors. In the right half we present results from the baseline (gray), where the left column is generated and the next three columns to its right are its three nearest-neighbors.