Learning Representations and Generative Models for 3D Point Clouds – Supplementary Material –

Panos Achlioptas [1] **Olga Diamanti** [1] **Ioannis Mitliagkas** [2] **Leonidas Guibas** [1]

1. AE Details

The encoding layers of our AEs were implemented as 1D-convolutions with ReLUs, with kernel size of 1 and stride of 1, i.e. treating each 3D point independently. Their decoding layers, were MLPs built with FC-ReLUs. We used Adam (Kingma & Ba, 2014) with initial learning rate of 0.0005, β_1 of 0.9 and a batch size of 50 to train all AEs.

1.1. AE used for SVM-based experiments

For the AE mentioned in the SVM-related experiments of Section 5.1 of the main paper, we used an encoder with $128, 128, 256$ and 512 filters in each of its layers and a decoder with $1024, 2048, 2048 \times 3$ neurons, respectively. Batch normalization was used between every layer. We also used online data augmentation by applying random rotations along the gravity-(z)-axis to the input point clouds of each batch. We trained this AE for 1000 epochs with the CD loss and for 1100 with the EMD.

1.2. All other AEs

For all other AEs, the encoder had $64, 128, 128, 256$ and k filters at each layer, with k being the bottle-neck size. The decoder was comprised by 3 FC-ReLU layers with $256, 256, 2048 \times 3$ neurons each. We trained these AEs for a maximum of 500 epochs when using single class data and 1000 epochs for the experiment involving 5 shape classes (end of Section 5.2, main paper).

1.3. AE regularization

To determine an appropriate size for the latent-space, we constructed 8 (otherwise architecturally identical) AEs with bottleneck sizes $k \in \{4, 8 \dots, 512\}$ and trained them with point clouds of the chair object class, under the two losses

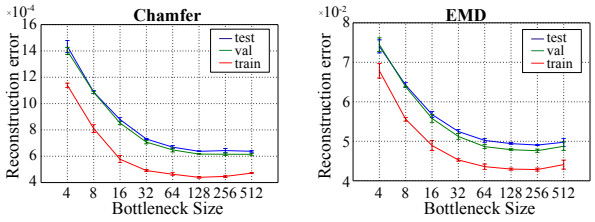

Figure 1. The bottleneck size was fixed at 128 in all single-class experiments by observing the reconstruction loss of the AEs, shown here for various bottleneck sizes, when training with the data of the chair class.

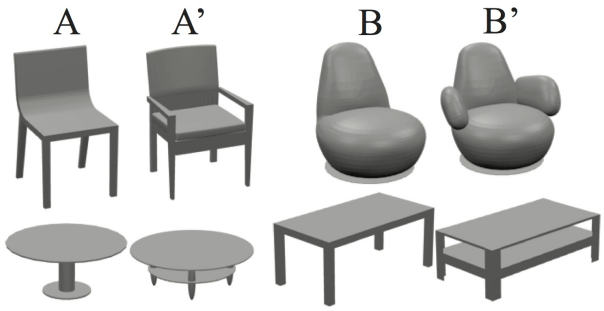

Figure 2. Shape Analogies using our learned representation. Shape B' relates to B in the same way that shape A' relates to A.

(Fig. 1). We repeated this procedure with pseudo-random weight initializations three times and found that $k = 128$ had the best generalization error on the test data, while achieving minimal reconstruction error on the train split.

Remark. Different AE setups brought no noticeable advantage over our main architecture. Concretely, adding drop-out layers resulted in worse reconstructions and using batch-norm on the encoder *only*, sped up training and gave us slightly better generalization error when the AE was trained with single-class data. Exclusively, for the SVM experiment of Section 5.1 of the main paper we randomly rotate the input chairs to promote latent features that are rotation-invariant.

[1]Department of Computer Science, Stanford University, USA [2]Department of Computer Science and Operations Research, University of Montréal, Canada. Correspondence to: Panos Achlioptas <optas@cs.stanford.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

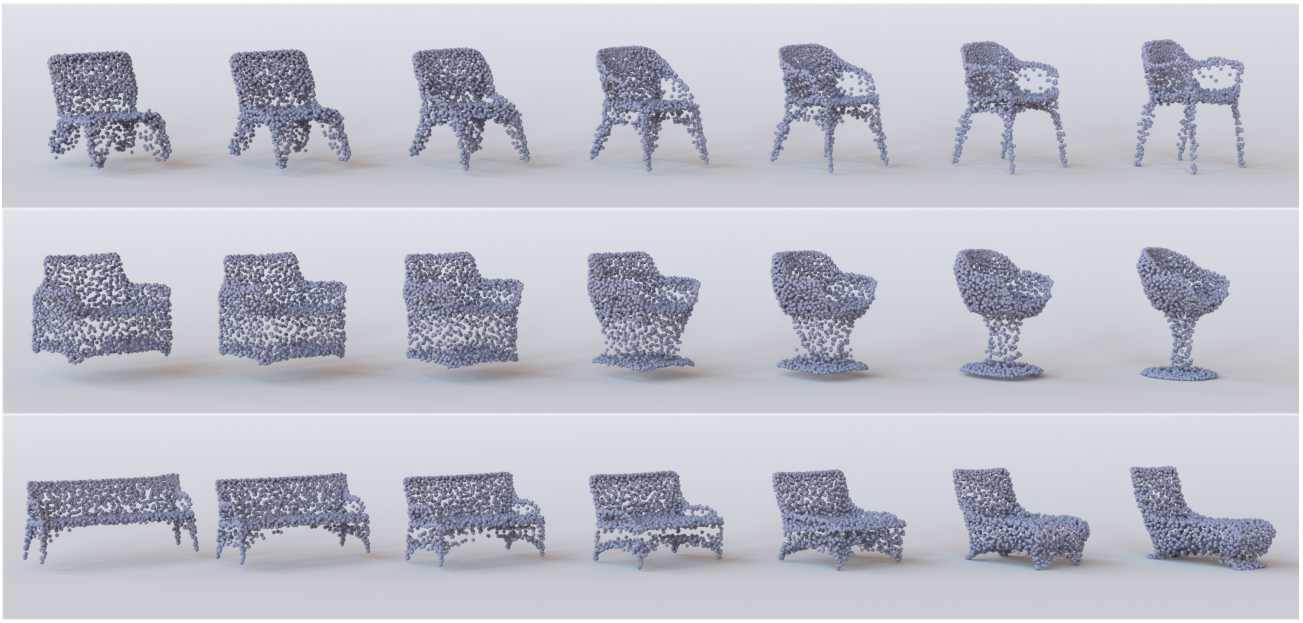

Figure 3. Interpolating between different point clouds (left and right-most of each row), using our latent space representation. Note the interpolation between structurally and topologically different shapes. **Note**: for all our illustrations that portray point clouds we use the Mitsuba renderer (Jakob, 2010).

2. Applications of the Latent Space Representation

For shape editing applications, we use the embedding we learned with the AE-EMD trained *across all* 55 object classes, not separately per-category. This showcases its ability to encode features for different shapes, and enables interesting applications involving different kinds of shapes.

Editing shape parts. We use the shape annotations of Yi et al.(Yi et al., 2016) as guidance to modify shapes. As an example, assume that a given object category (e.g. chairs) can be further subdivided into two sub-categories $\mathcal{A}$ and $\mathcal{B}$: every object $A \in \mathcal{A}$ possesses a certain structural property (e.g. has armrests, is four-legged, etc.) and objects $B \in \mathcal{B}$ do not. Using our latent representation we can model this structural difference between the two sub-categories by the difference between their average latent representations $\mathbf{x}_\mathcal{B} - \mathbf{x}_\mathcal{A}$, where $\mathbf{x}_\mathcal{A} = \sum_{A \in \mathcal{A}} \mathbf{x}_A$, $\mathbf{x}_\mathcal{B} = \sum_{B \in \mathcal{B}} \mathbf{x}_B$. Then, given an object $A \in \mathcal{A}$, we can change its property by transforming its latent representation: $x_{A'} = x_A + \mathbf{x}_\mathcal{B} - \mathbf{x}_\mathcal{A}$, and decode $\mathbf{x}_{\mathcal{A}'}$ to obtain $A' \in \mathcal{B}$. This process is shown in Fig. 3 of the main paper.

Interpolating shapes. By linearly interpolating between the latent representations of two shapes and decoding the result we obtain intermediate variants between the two shapes. This produces a "morph-like" sequence with the two shapes at its end points Fig. 2 of main paper and Fig. 3 here). Our latent representation is powerful enough to support removing and merging shape parts, which enables morphing between shapes of significantly different appearance. Our cross-category latent representation enables morphing between shapes of different classes, cfg. the second row for an interpolation between a bench and a sofa.

Shape analogies. Another demonstration of the Euclidean nature of the latent space is demonstrated by finding analogous shapes by a combination of linear manipulations and Euclidean nearest-neighbor searching. Concretely, we find the difference vector between A and A', we add it to shape B and search in the latent space for the nearest-neighbor of that result, which yields shape B'. We demonstrate the finding in Fig. 2 with images taken from the meshes used to derive the underlying point clouds to help the visualization. Finding shape analogies has been of interest recently in the geometry processing community (Rustamov et al., 2013; Huang et al., 2018).

Loss	ModelNet40			ModelNet10		
	C-plt	icpt	loss	C-plt	icpt	loss
EMD	0.09	0.5	hng	0.02	3	sq-hng
CD	0.25	0.4	sq-hng	0.05	0.2	sq-hng

Table 1. Training parameters of SVMs used in each dataset with each structural loss of the AE. *C-penalty (C-plt)*: term controlling the trade-off between the size of the learned margin and the misclassification rate; *intercept (icpt)*: extra dimension appended on the input features to center them; *loss*: svm's optimization loss function: hinge (*hng*), or squared-hinge (*sq-hng*).

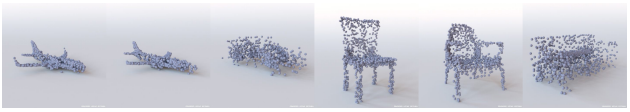

Figure 4. Synthetic results produced by the r-GAN. From left to right: airplanes, car, chairs, sofa.

3. Autoencoding Human Forms

In addition to ShapeNet core which contains man-made only objects, we have experimented with the D-FAUST dataset of (Bogo et al., 2017) that contains meshes of human subjects. Specifically, D-FAUST contains 40K scanned meshes of 10 human subjects performing a variety of motions. Each human performs a set of (maximally) 14 motions, each captured by a temporal sequence of ∼300 meshes. For our purposes, we use a random subset of 80 (out of the 300) meshes for each human/motion and extract from each mesh a point cloud with 4096 points. Our resulting dataset contains a total of 10240 point clouds and we use a train-test-val split of $[70\%, 20\%, 10\%]$ - while enforcing that every split contains *all* human/motion combinations. We use this data to train and evaluate an AE-EMD that is identical to the single-class AE presented in the main paper, with the only difference being the number of neurons of the last layer (4096×3 instead of 2048×3).

We demonstrate reconstruction and interpolation results in Figs. 5 and 6. For a given human subject and a specific motion we pick at random two meshes corresponding to time points t_0, t_1 (with $t_1 > t_0$) and show their reconstructions along with the ground truth in Fig. 5 (left-most and right-most of each row). In the same figure we also plot the reconstructions of two random meshes captured in (t_0, t_1) (middle-two of each row). In Fig. 6, instead of encoding/decoding the ground truth test data, we show decoded linear *interpolations* between the meshes of t_0, t_1.

Figure 5. Reconstructions of unseen shapes from the *test* split extracted from the D-FAUST dataset of (Bogo et al., 2017) with an AE-EMD decoding point clouds with 4096 points. In each row the poses depict a motion (left-to-right) as it progress in time.

4. Shape Completions

An important application that our AE architecture can be used for is that of completing point clouds that contain limited information of the underlying geometry. Typical range scans acquired for an object in real-time can often miss entire regions of the object due to the existence of self-occlusions and the lack of adequate (or "dense") view-point registrations. This fact makes any sensible solution to this problem of high practical importance. To address it here, we resort in a significantly different dataset than the ones used in the rest of this paper. Namely, we utilize the dataset of (Dai et al., 2016) that contains pairs of complete (intact) 3D CAD models and *partial* versions of them. Specifically, for each object of ShapeNet (core) it contains six partial point clouds created by the aggregation of frames taken over a limited set of view-points in a virtual trajectory established around the object. Given this data, we first fix the dimensionality of the partial point clouds to be 2048 points for each one by randomly sub-sampling them. Second, we apply uniform-in-area sampling to each complete CAD model to extract from it 4096 points to represent a "complete" ground-truth datum. All the resulting point clouds are centered in the unit-sphere and (within a class) the partial and complete point clouds are co-aligned. Last, we train class-specific neural-nets with Chair, Table and Airplane data and a train/val/test split of [80%, 5%, 15%].

4.1. Architecture

The high level design of the architecture we use for shape-completions is identical to the AE, i.e. independent-convolutions followed by FCs, trained under a structural loss (CD or EMD). However, essential parts of this network are different: depth, bottleneck size (controlling compression ratio) and the crucial differentiation between the input

Figure 6. Interpolating between different point clouds from the *test* split (left and right-most of each row) of the D-FAUST dataset of (Bogo et al., 2017). These linear interpolations have captured some of the dynamics of the corresponding motions: 'chicken-wings' (first row), 'shake shoulders' (second row) and 'jumping jacks' (third row). Compare to Fig.5 that contains ground-truth point clouds in the same time interval.

and the output data. Technically, the resulting architecture is an Abstractor-Predictor (AP) and is comprised by three layers of independent per-point convolutions, with filter sizes of $[64, 128, 1024]$, followed by a max-pool, which is followed by an FC-ReLU (1024 neurons) and a final FC layer (4096×3 neurons). We don't use batch-normalization between any layer and train each class-specific AP for a maximum of 100 epochs, with ADAM, initial learning rate of 0.0005 and a batch size of 50. We use the minimal per the validation split model (epoch) for evaluating our models with the test data.

4.2. Evaluation

We use the specialized point cloud completion metrics introduced in (Sung et al., 2015). That is a) the *accuracy*: which is the fraction of the predicted points that are within a given radius (ρ) from any point in the ground truth point cloud and b) the *coverage*: which is the fraction of the ground-truth points that are within ρ from any predicted point. In Table 2 we report these metrics (with a $\rho = 0.02$ similarly to (Sung et al., 2015)) for class-specific networks that were trained with the EMD and CD losses respectively. We observe that the CD loss gives rise to more accurate but also less complete outputs, compared to the EMD. This highlights again the greedy nature of CD – since it does not take into account the matching between input/output, it can get generate completions that are more concentrated around the (incomplete) input point cloud. Figure 7 shows the corresponding completions of those presented in the main paper, but with a network trained under the CD loss.

Class	Airplane	Chair	Table
Test-size	4.5K	6K	6K
Acc-CD	**96.9**	**86.5**	**87.6**
Acc-EMD	94.7	77.1	78.4
Cov-CD	96.6	77.5	75.2
Cov-EMD	**96.8**	**82.6**	**83.0**

Table 2. Performance of point cloud *completions* on ShapeNet *test* data. Comparison between Abstractor-Predictors trained under the CD or EMD losses, on mean **Acc**uracy and **Cov**erage, across each class. The size of each test-split is depicted in the first row.

5. SVM Parameters for Auto-encoder Evaluation

For the classification experiments of Section 5.1 (main paper) we used a one-versus-rest linear SVM classifier with an l_2 norm penalty and balanced class weights. The exact optimization parameters can be found in Table 1. The confusion matrix of the classifier evaluated on our latent codes on ModelNet40 is shown in Fig. 8.

6. r-GAN Details

The discriminator's first 5 layers are 1D-convolutions with stride/kernel of size 1 and $\{64, 128, 256, 256, 512\}$ filters each; interleaved with leaky-ReLU. They are followed by a feature-wise max-pool. The last 2 FC-leaky-ReLU lay-

Figure 7. Point cloud *completions* of a network trained with partial and complete (input/output) point clouds and the **CD loss**. Each triplet shows the partial input from the test split (left-most), followed by the network's output (middle) and the complete ground-truth (right-most). Also compare with Fig. 4 of main paper that portrays the corresponding completions of a network trained with the EMD loss.

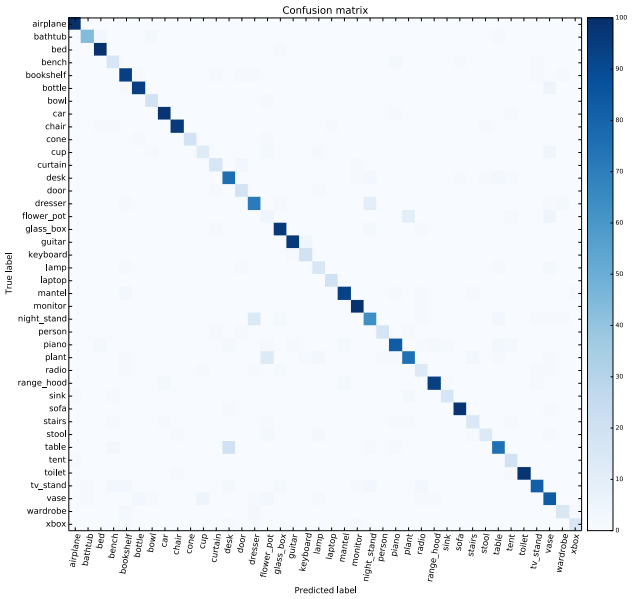

Figure 8. Confusion matrix for the SVM-based classification of Section 5.1, for the Chamfer loss on ModelNet40. The class pairs most confused by the classifier are dresser/nightstand, flower pot/plant. Better viewed in the electronic version.

ers have $\{128, 64\}$, neurons each and they lead to single sigmoid neuron. We used 0.2 units of leak.

The generator consists of 5 FC-ReLU layers with $\{64, 128, 512, 1024, 2048 \times 3\}$ neurons each. We trained r-GAN with Adam with an initial learning rate of 0.0001, and $beta_1$ of 0.5 in batches of size 50. The noise vector was drawn by a spherical Gaussian of 128 dimensions with zero mean and 0.2 units of standard deviation.

Some synthetic results produced by the r-GAN are shown in Fig. 4.

7. l-GAN Details

The discriminator consists of 2 FC-ReLU layers with $\{256, 512\}$ neurons each and a final FC layer with a single sigmoid neuron. The generator consists of 2 FC-ReLUs with $\{128, k = 128\}$ neurons each. When used the l-Wasserstein-GAN, we used a gradient penalty regularizer $\lambda = 10$ and trained the critic for 5 iterations for each training iteration of the generator. The training parameters (learning rate, batch size) and the generator's noise distribution were the same as those used for the r-GAN.

8. Model Selection of GANs

All GANs are trained for maximally 2000 epochs; for each GAN, we select one of its training epochs to obtain the "final" model, based on how well the synthetic results match the ground-truth distribution. Specifically, at a given epoch, we use the GAN to generate a set of synthetic point clouds, and measure the distance between this set and the validation set. We avoid measuring this distance using MMD-EMD, given the high computational cost of EMD. Instead, we use either the JSD or MMD-CD metrics to compare the synthetic dataset to the validation dataset. To further reduce the computational cost of model selection, we only check every 100 epochs (50 for r-GAN). The generalization error of the various GAN models, at various training epochs, as measured by MMD and JSD is shown in Fig. 9 (left and middle).

Using the same JSD criterion, we also select the number and covariance type of Gaussian components for the GMM (Fig. 10, left), and obtain the optimal value of 32 components. GMMs performed much better with full (as opposed to diagonal) covariance matrices, suggesting strong correlations between the latent dimensions (Fig. 10, right).

When using MMD-CD as the selection criterion, we obtain models of similar quality and at similar stopping epochs (see Table 3); the optimal number of Gaussians in this case was 40. The training behavior measured using MMD-CD can be seen in Fig. 9 (right).

9. Voxel AE Details

Our voxel-based AEs are fully-convolutional with the encoders consisting of 3D-Conv-ReLU layers and the decoders of 3D-Conv-ReLU-transpose layers. Below, we list the parameters of consecutive layers, listed left-to-right. The layer parameters are denoted in the following manner: (number of filters, filter size). Each Conv/Conv-Transpose has a stride of 2 except the last layer of the 32^3 decoder which has 4. In the last layer of the decoders we do not use a non-linearity. The abbreviation "bn" stands for batch-normalization.

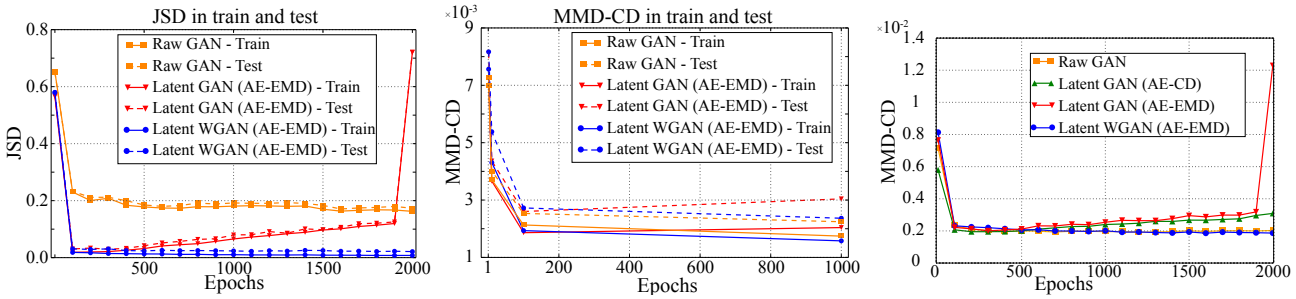

Figure 9. Left/middle: Generalization error of the various GAN models, at various training epochs. Generalization is estimated using the JSD (left) and MMD-CD (middle) metrics, which measure closeness of the synthetic results to the training resp. test ground truth distributions. The plots show the measurements of various GANs. Right: Training trends in terms of the MMD-CD metric for the various GANs. Here, we sample a set of synthetic point clouds for each model, of size 3x the size of the ground truth test dataset, and measure how well this synthetic dataset matches the ground truth in terms of MMD-CD. This plot complements Fig. 6 (left) of the main paper, where a different evaluation measure was used - note the similar behavior.

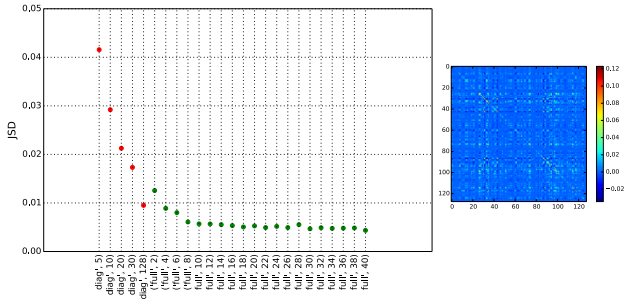

Figure 10. GMM model selection. GMMs with a varying number of Gaussians and covariance type are trained on the latent space learned by and AE trained with EMD and a bottleneck of 128. Models with a full covariance matrix achieve significantly smaller JSD than models trained with diagonal covariance. For those with full covariance, 30 or more clusters seem sufficient to achieve minimal JSD. On the right, the values in a typical covariance matrix of a Gaussian component are shown in pseudocolor - note the strong off-diagonal components.

- 32^3 - model
  Encoder: Input $\rightarrow$ (32, 6) $\rightarrow$ (32, 6) $\rightarrow$ bn $\rightarrow$ (64, 4) $\rightarrow$ (64, 2) $\rightarrow$ bn $\rightarrow$ (64, 2)
  Decoder: (64, 2) $\rightarrow$ (32, 4) $\rightarrow$ bn $\rightarrow$ (32, 6) $\rightarrow$ (1, 8) $\rightarrow$ Output

- 64^3 - model
  Encoder: Input $\rightarrow$ (32, 6) $\rightarrow$ (32, 6) $\rightarrow$ bn $\rightarrow$ (64, 4) $\rightarrow$ (64, 4) $\rightarrow$ bn $\rightarrow$ (64, 2) $\rightarrow$ (64, 2)
  Decoder: (64, 2) $\rightarrow$ (32, 4) $\rightarrow$ bn $\rightarrow$ (32, 6) $\rightarrow$ (32, 6) $\rightarrow$ bn $\rightarrow$ (32, 8) $\rightarrow$ (1, 8) $\rightarrow$ Output

We train each AE for 100 epochs with Adam under the binary cross-entropy loss. The learning rate was 0.001, the β_1 0.9 and the batch size 64. To validate our voxel AE architectures, we compared them in terms of reconstruction quality to the state-of-the-art method of (Tatarchenko et al., 2017) and obtained comparable results, as demonstrated in Table 4.

Method	Epoch	JSD	MMD-CD	MMD-EMD	COV-EMD	COV-CD
A	1350	0.1893	0.0020	0.1265	19.4	54.7
B	300	0.0463	0.0020	0.0800	32.6	58.2
C	200	0.0319	0.0022	0.0684	57.6	58.7
D	1700	0.0240	0.0020	0.0664	64.2	64.7
E	-	**0.0182**	**0.0018**	**0.0646**	**68.6**	**69.3**

Table 3. Evaluation of five generators on **test-split** of *chair* data on epochs/models that were selected via minimal MMD-CD on the validation-split. We report: A: r-GAN, B: l-GAN (AE-CD), C: l-GAN (AE-EMD) , D: l-WGAN (AE-EMD), E: GMM-40-F (AE-EMD). GMM-40-F stands for a GMM with 40 Gaussian components with full covariances. The reported scores are averages of three pseudo-random repetitions. Compare this with Table 3 of the main paper. Note that the overall quality of the selected models remains the same, irrespective of the metric used for the selection.

10. Memorization Baseline

Here we compare our GMM-generator against a model that memorizes the training data of the chair class. To do this, we either consider the entire training set or randomly subsample it, to create sets of different sizes. We then evaluate our metrics between these memorized sets and the point clouds of test split (see Table 5). The coverage/fidelity obtained by our generative models (last row) is slightly lower than the equivalent in size case (third row) as expected: memorizing the training set produces good coverage/fidelity with respect to the test set when they are both drawn from the same population. This speaks for the validity of our metrics. Naturally, the advantage of using a learned representation lies in learning the structure of the underlying

Voxel Resolution	32	64
Ours	92.7	88.4
(Tatarchenko et al., 2017)	93.9	90.4

Table 4. Reconstruction quality statistics for our dense voxel-based AE and the one of (Tatarchenko et al., 2017) for the ShapeNetCars dataset. Both approaches use a 0.5 occupancy threshold and the train-test split of (Tatarchenko et al., 2017). Reconstruction quality is measured by measuring the intersection-over-union between the input and synthesized voxel grids, namely the ratio between the volume in the voxel grid that is 1 in both grids divided by the volume that is 1 in at least one grid.

Sample Set Size	COV-CD	MMD-CD	COV-EMD	MMD-EMD
Entire —Train—	97.3	0.0013	98.2	0.0545
$1\times$ —Test—	54.0	0.0023	51.9	0.0699
$3\times$ —Test—	79.4	0.0018	78.6	0.0633
Full-GMM/32 ($3\times$ —Test—)	68.9	0.0018	67.4	0.0651

Table 5. Quantitative results of a baseline sampling/memorizing model, for different sizes of sets sampled from the training set and evaluated against the test split. The first three rows show results of a memorizing model, while the third row corresponds to our generative model. The first row shows the results of memorizing the entire training chair dataset. The second and third rows show the averages of three repetitions of the sub-sampling procedure with different random seeds.

space instead of individual samples, which enables compactly representing the data and generating novel shapes as demonstrated by our interpolations. In particular, note that while some mode collapse is present in our generative results, as indicated by the $\sim 10\%$ drop in coverage, the MMD of our generative models is almost identical to that of the memorization case, indicating excellent fidelity.

11. More Comparisons with Wu et al.

In addition to the EMD-based comparisons in Table 4 of the main paper, in Tables 6, 7 we provide comparisons with (Wu et al., 2015) for the ShapeNet classes for which the authors have made publicly available their models. In Table 6 we provide JSD-based comparisons for two of our models. In Table 7 we provide Chamfer-based Fidelity/Coverage comparisons on the *test* split, that complement the EMD-based ones of Table 4 in the main paper.

Comparisons on training data. In Table 8 we compare to (Wu et al., 2016) in terms of the JSD and MMD-CD on the training set of the *chair* category. Since (Wu et al., 2016) do not use any train/test split, we perform 5 rounds of sampling 1K synthetic results from their models and report the best values of the respective evaluation metrics.

Class	A	B		C	
	Tr+Te	Tr	Te	Tr	Te
airplane	-	0.0149	0.0268	**0.0065**	0.0191
car	0.1890	0.0081	0.0109	**0.0063**	0.0108
rifle	0.2012	0.0212	0.0364	**0.0092**	0.0214
sofa	0.1812	**0.0102**	0.0102	**0.0102**	0.0101
table	0.2472	0.0058	0.0177	**0.0035**	0.0143

Table 6. JSD-based comparison between A: (Wu et al., 2016) and our generative models – B: a latent GAN, C: our GM with 32 full-covariance Gaussian components. Both B and C were trained on the latent space of our AE with the EMD structural loss. Note that the l-GAN here uses the same "vanilla" adversarial objective as (Wu et al., 2016). Tr: training split, Te: testing split.

Class	MMD-CD		COV-CD	
	A	B	A	B
airplane	-	0.0005	-	71.1
car	0.0015	**0.0007**	22.9	**63.0**
rifle	0.0008	**0.0005**	56.7	**71.7**
sofa	0.0027	**0.0013**	42.40	**75.5**
table	0.0058	**0.0016**	16.7	**71.7**

Table 7. CD based MMD and Coverage comparison between A: Wu et al. (2016) and B: our generative model on the *test* split of each class. Our generative model is a GM with 32 full-covariance Gaussian components, trained on the latent space of our AE with the EMD structural loss. Note that Wu et al. used *a*ll models of each class for training.

We also report the average classification probability of the synthetic samples to be classified as chairs by the PointNet classifier. The r-GAN mildly outperforms (Wu et al., 2016) in terms of its diversity (as measured by JSD/MMD), while also creating realistic-looking results, as shown by the classification score. The l-GANs perform even better, both in terms of classification and diversity, with less training epochs. Finally, note that the PointNet classifier was trained on ModelNet, and (Wu et al., 2016) occasionally generates shapes that only rarely appear in ModelNet. In conjunction with their higher tendency for mode collapse, this partially accounts for their lower classification scores.

12. Limitations

Figure 11 shows some failure cases of our models. Chairs with rare geometries (left two images) are sometimes not faithfully decoded. Additionally, the AEs may miss high-frequency geometric details, e.g. a hole in the back of a chair (middle), thus altering the style of the input shape. Finally, the r-GAN often struggles to create realistic-looking shapes (right) – while the r-GAN chairs are easily visually recog-

Metric	A	B	C	D	E	F
JSD	0.1660	0.1705	0.0372	0.0188	0.0077	**0.0048**
MMD-CD	0.0017	0.0042	0.0015	0.0018	0.0015	**0.0014**
CLF	84.10	87.00	96.10	94.53	89.35	87.40

Table 8. Evaluating six generators on **train-split** of *chair* dataset on epochs/models selected via minimal JSD on the validation-split. We report: A: r-GAN, B: (Wu et al., 2016) (a volumetric approach), C: l-GAN(AE-CD), D: l-GAN(AE-EMD), E: l-WGAN(AE-EMD), F: GMM(AE-EMD). Note that the average classification score attained by the ground-truth point clouds was 84.7%.

Figure 11. The AEs might fail to reconstruct uncommon geometries or might miss high-frequency details: first four images - left of each pair is the input, right the reconstruction. The r-GAN may synthesize noisy/unrealistic results, cf. a car (right most image).

nizable, it has a harder time on cars. Designing more robust raw-GANs for point clouds remain an interesting avenue for future work. A limitation of our shape-completion pipeline regards the style of the partial shape, which might not be well preserved in the completed point cloud (see Fig. 13 for an example).

References

Bogo, F., Romero, J., Pons-Moll, G., and Black, M. J. Dynamic FAUST: Registering human bodies in motion. In *IEEE CVPR*, 2017.

Dai, A., Qi, C. R., and Nießner, M. Shape completion using 3d-encoder-predictor cnns and shape synthesis. *http://arxiv.org/abs/1612.00101*, 2016.

Huang, R., Achlioptas, P., Guibas, L., and Ovsjanikov, M. Latent space representation for shape analysis and learning. *http://arxiv.org/abs/1806.03967*, 2018.

Jakob, W. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Rustamov, R. M., Ovsjanikov, M., Azencot, O., Ben-Chen, M., Chazal, F., and Guibas, L. Map-based exploration of intrinsic shape differences and variability. *ACM Trans. Graph.*, 32(4), July 2013.

Sung, M., Kim, V. G., Angst, R., and Guibas, L. J. Data-driven structural priors for shape completion. *ACM Transactions on Graphics (TOG)*, 34(6):175, 2015.

Tatarchenko, M., Dosovitskiy, A., and Brox, T. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. *CoRR*, abs/1703.09438, 2017.

Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *NIPS*. 2016.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In *IEEE CVPR*, 2015.

Yi, L., Kim, V. G., Ceylan, D., Shen, I., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., and Guibas, L. J. A scalable active framework for region annotation in 3d shape collections. *ACM Trans. Graph.*, 35(6), 2016.

Figure 12. The 32 centers of the GMM fitted to the latent codes, and decoded using the decoder of the AE-EMD.

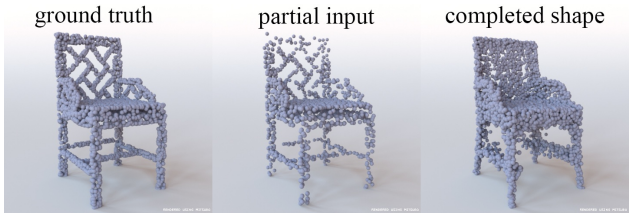

Figure 13. Our completion network might fail to preserve some of the style information in the partial point cloud, even though a reasonable shape is produced.