LEARNING REPRESENTATIONS AND GENERATIVE MODELS FOR 3D POINT CLOUDS

Anonymous authors
Paper under double-blind review

ABSTRACT

yianni: update this to reflect more the new stuff Three-dimensional geometric data offer an excellent domain for studying representation learning and generative modeling. In this paper, we look at geometric data represented as point clouds. We introduce a deep autoencoder network for point clouds, which outperforms the state of the art in 3D recognition tasks. We also design GAN architectures to generate novel point-clouds. Importantly, we show that by training the GAN in the latent space learned by the autoencoder, we greatly boost the GAN's data-generating capacity, creating significantly more diverse and realistic geometries, with far simpler architectures. The expressive power of our learned embedding, obtained without human supervision, enables basic shape editing applications via simple algebraic manipulations, such as semantic part editing, shape analogies and shape interpolation.

1 INTRODUCTION

[3D data are très cool] Three-dimensional (3D) representations of real-life objects are a core tool for vision, robotics, medicine and augmented and virtual reality applications. [Classic encodings are handy but lack semantics] Classic encodings include view-based projections (Su et al., 2015; Wei et al., 2016; Kalogerakis et al., 2016), volumetric grids (Qi et al., 2016c; Wu et al., 2015) and graphs (Bruna et al., 2013; Henaff et al., 2015; Defferrard et al., 2016; Yi et al., 2016b). **todo: The citations above are too modern for the point we are making here. We need older citations for view-based, graph and other representations.** These encodings are effective when used for acquisition and rendering, however they are poor in semantics. For example, naïvely interpolating between two different cars in a view-based representation does not yield a representation of an 'intermediate' car. Furthermore, these raw representations are not well suited for the design of generative models using classic statistical methods. [Modeling attempts] Some computational geometrists have managed to encode human intuition into carefully designed parametric models () enabling semantics and generation. However, those approaches only generate models of limited scope and require serious design work by experts.

[Deep learning] Recent advances in deep learning bring the promise of a *data-driven approach*. In domains where data is plentiful, deep learning tools have eliminated the need for hand-crafting features and models. This advent not only opens up certain fields to non-domain experts, but importantly, accelerates scientific experiementation and progress. Experts are free to focus on more creative aspects of their work than just manually modeling obvious human intuition about the world. Deep learning architectures like autoencoders (AEs) (Rumelhart et al., 1988; Kingma & Welling, 2013) and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014; Radford et al., 2015; Denton et al., 2015; Che et al., 2016) are successful at learning complex data representations and generating realistic samples from the underlying distributions. Applied predominantly on natural image synthesis (Denton et al., 2015; Ledig et al., 2016; Nguyen et al., 2016) and unsupervised learning (Radford et al., 2015) with impressive results, these architectures are the object of intense research in the machine learning community. [Deep learning for 3D] Recently, deep learning architectures for view-based projections (Su et al., 2015; Wei et al., 2016; Kalogerakis et al., 2016), volumetric grids (Qi et al., 2016c; Wu et al., 2015) and graphs (Bruna et al., 2013; Henaff et al., 2015; Defferrard et al., 2016; Yi et al., 2016b) have started appearing in computational geometry literature.

[Point-clouds new in DL] In this paper we focus on point clouds: a relatively unexplored 3D modality. Point clouds provide a homogeneous, expressive and compact representation of surface geometry, easily amenable to geometric operations. These properties make them attractive from a learning point of view. In addition, they come up as the output of common range-scanning acquisition pipelines used in devices like the Kinect and iPhone's recent face identification feature. Only a handful of deep architectures for 3D point clouds exist in the literature: PointNet (Qi et al., 2016a) successfully tackled classification and segmentation tasks; Kalogerakis et al. (2016) used point-clouds as an intermediate step in their pipeline **todo: are we missing other deep work on point clouds? see Skype where I sent to you two papers**. We provide the first results that use deep architectures to learn representations and generative models for point clouds.

[Generative modeling] Generative models have gathered increased attention recently in the deep learning community with the introduction of GANs (Goodfellow et al., 2014). An issue with GAN-based generative pipelines is that training them is notoriously hard and unstable (Salimans et al., 2016). More importantly, *there is no generally accepted way to evaluate generative models*. A number of methods have been proposed (**??**), but they all have their drawbacks (**?**). In evaluating generative models one is interested in both *fidelity*, i.e. how much generated points resemble the actual data, and *coverage*, i.e. what fraction of the data distribution a generated sample represents. The latter is especially important given the tendency of certain GANs to exhibit mode collapse. Recent work uses clever application of basic probability to estimate *the number of different points generated* (**?**). However, this counting measure betrays no information about the topology of misses. In particular it assigns the same penalty to models that miss individual examples interspersed all over and models that miss whole contiguous pieces of the data manifold. We provide simple methods to deal with both the training and evaluation issue in our target domain.

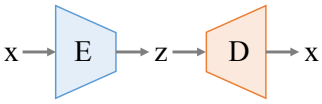
Our contributions

- We design a new AE architecture—inspired by recent architectures used for classification (Qi et al., 2016b)— that is capable of learning compact representations of point clouds with excellent reconstruction quality even on unseen samples. The learned representations are (i) good for classification via simple methods (SVM), improving on the state of the art (Wu et al., 2016); (ii) suitable for meaningful interpolations and semantic operations.
- We create the first set of generative models capable of (i) generating point clouds measurably similar to the training data and held-out test data; (ii) provide good coverage of the training and test dataset. We argue that jointly learning the representation and training the GAN is unnecessary for our modality. We propose a workflow that first learns a representation by training an AE with a compact bottleneck layer, then trains a plain GAN in that fixed latent representation. We point to theory (Arjovsky & Bottou, 2017) that supports this idea, and verify it empirically. Latent GANs are *much easier* to train than monolithic (raw) GANS and achieve superior reconstruction with much better coverage. Somewhat surprisingly, GMMs trained in the latent space of fixed AEs achieve the best performance across the board.
- To support our qualitative evaluation, we perform a careful study of various old and new metrics. We study various point cloud metrics (JSD on pointspanos: I don't like the "points" here, instead? discritized 3D space, Chamfer on clouds, EMD on clouds) and their ability to (i) be used as objectives for learning good representations; (ii) be used for the evaluation of generated samples. We propose a simple metric based on a minimum matching that can be used as a measure of coverage. We find that Chamfer distances fail to discriminate certain pathological cases from good examples.
- We show that training multi-class GANs works almost on par with GAN-per-category, in the latent space.

The rest of this paper is organized as follows **todo: mention sections**

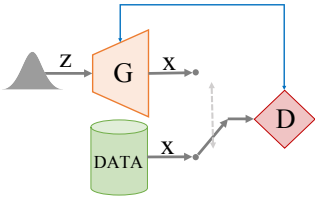
2 BACKGROUND

In this section we give some background and introduce the basic building blocks for our work.



Autoencoders. Autoencoders (AE - inset) are deep architectures that aim to reproduce their input. They are especially useful, when they contain a narrow *bottleneck layer* between input and output. Upon successful training, the bottleneck layer corresponds to a low-dimensional representation, a *code* for the dataset. The Encoder (E) learns to compress a data point $\boldsymbol{x}$ into its latent representation, $\boldsymbol{z}$. The Decoder (D) can then reproduce $\boldsymbol{x}$ from its encoded version $\boldsymbol{z}$.

yianni: shall we keep the figure of the AE? of the GAN?



Generative Adversarial Networks. GANs are the state of the art in generative models. The basic architecture (inset) is based on a adversarial game between a *generator* (G) and a *discriminator* (D). The generator aims to synthesize samples that look indistinguishable from real data (drawn from $\boldsymbol{x} \sim p_{\text{data}}$) by passing a randomly drawn sample $\boldsymbol{z} \sim p_z$ through the generator function G. The discriminator tries to tell synthesized from real samples. The most commonly used losses for the discriminator and generator networks are:

$$J^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) = -\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}\sim p_z} \log \left(1 - D\left(G(\boldsymbol{z})\right)\right) , \quad (1)$$

$$J^{(G)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) = -\mathbb{E}_{\boldsymbol{z}\sim p_z} \log D(G(\boldsymbol{z})) , \quad (2)$$

where $\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}$ are the parameters for the discriminator and the generator network respectively.

yianni: Mention Wasserstein-based here?

Challenges specific to point cloud geometry. Point clouds as an input modality present a unique set of challenges when building a network architecture. As an example, the convolution operator – now ubiquitous in image-processing pipelines – requires the signal (in our case, geometry) to be defined on top of an underlying grid-like structure. Such a structure is not available in raw point clouds, which renders them significantly more difficult to encode than e.g. images or voxel grids. Recent classification work on point clouds (PointNet – Qi et al. (2016b)) bypasses this issue by circumventing 2D convolutions. Another issue with point clouds as a representation is that they are unordered - any permutation of a point set still describes the same shape. This complicates comparisons between two point sets, typically needed to define a loss function. This unorderedness of point clouds also creates the need for making the encoded feature permutation invariant.

Point-set distances. Two permutation-invariant metrics for comparing unordered point sets have been proposed in the literature Fan et al. (2016). On the one hand, the *Earth Mover's* distance (EMD) Rubner et al. (2000) is the solution of a transportation problem which attempts to transform one set to the other. For two equally sized subsets $S_1 \subseteq R^3, S_2 \subseteq R^3$, their EMD is defined by $d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2$. where ϕ is a bijection. Interpreted as a loss, EMD is differentiable almost everywhere. On the other hand, the (still differentiable but more computationally efficient) *Chamfer* (pseudo)-distance (CD) measures the distance between each point in one set to its nearest neighbor in the other set, $d_{CH}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$.

Evaluation Metrics for representations and generative models. In the remainder of the paper, we will frequently be needing to compare a given set (distribution) of points clouds, whether reconstructed or synthesized, to its ground truth counterpart. For example, one might want to assess the quality of a representation model, in terms of how well it matches the ground truth distribution of point clouds. Alternatively, such a comparison might be done to evaluate the faithfulness and/or diversity of a generative model, and measure potential mode-collapse. To measure how well a point-cloud distribution A matches a ground truth distribution G, we introduce the following metrics:

MMD. Minimum Matching Distance. For every 3D point-cloud in A, we find its closest point-cloud in G; closeness can be computed using either CD or EMD, thus yielding two different metrics, MMD-CD and MMD-EMD. We report the average of the minimum distances over all matched shape pairs in the sets as the final value. A low MMD score indicates high fidelity of the elements of A to the ground truth distribution G, as it directly measures the quality of the pairwise matchings. yianni: see/scalp this

Coverage. For each point-cloud in A we find its closest neighbor in G; again, either of the structural distances can be used, yielding COV-CD and COV-EMD. Coverage is measured as the fraction of the point-clouds in G that were matched to point-clouds in A. Since coverage simply counts matched elements and ignores the quality of the matches, it is less sensitive to the fidelity of A to G; instead, a high coverage score typically indicates that most of G is roughly represented within A. yianni: see/scalp this

JSD. The Jensen Shannon Divergence of empirically estimated random variables defined in 3D space. Our training data are pre-aligned in 3D space, hence also the decoded and generated data; this allows measuring the degree to which the point clouds from the two distributions co-vary. To do this, we consider a 28^3 regularly-spaced voxel grid centered in the sphere of diameter 1. We construct a 28^3 random variable vector for A by counting the number of 3D points lying within each voxel, across all point clouds of A; similarly for G. We report the JSD of these two random vectors. yianni: mention if you like the "marginal" nature of JSD

3 Representation and Generative Models

In this section we describe the architectures of our representation and generative models for point clouds, starting from our autoencoder design. Later, we introduce a GAN architecture tailored to point-cloud data, followed by a more efficient pipeline that first trains an AE and trains a much smaller GAN in the learned latent space, and a simpler generative model based on Gaussian Mixtures.

3.1 Learning representations of 3D point clouds

The input to our AE network is a point cloud with 2048 points (2048×3 matrix), representing a 3D shape. The encoder architecture follows the principle of Qi et al. (2016b): mlp-layers with increasing feature size, ending with a "symmetric" function. This approach encodes every point independently and uses a permutation-invariant (symmetric) function to make a joint representation. In our implementation we use 5 mlp layers, each followed by a ReLU and a batch-norm layer. The last layer is passed to a feature-wise maximum to produce a k-dimensional vector which is the basis for our latent space. The decoder transforms the latent vector with 3 fully connected layers, the first two having ReLUs, to produce a 2048×3 output. To cope with the lack of order in the point cloud, we explore both the efficient EMD-distance approximation (Fan et al., 2016) and the Chamfer-Distance as our structural losses; this yields two distinct AE models, referred to as AE-EMD and AE-CD.

Different AE setups (denoising/regularised) brought no noticeable advantage over our “vanilla” architecture. Adding drop-out layers resulted in worse reconstructions and using batch-norm on the encoder only, speeded up training and gave us slightly better generalization error when the AE was trained with single-class data.

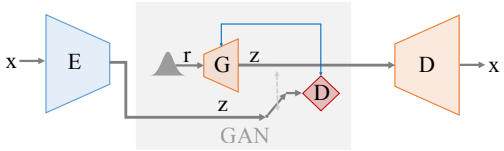
To determine an appropriate size for the latent-space, we constructed AEs differing only by $k \in 4, 8 \ldots, 512$ and trained them with point-clouds of a signle object class, under the two losses. We repeated this procedure with pseudo-random weight initializations three times (Figures olga: link to error-bar-plots) and found that $k = 128$ had the best generalization error on the test data, while achieving minimal reconstruction error on the the train split.

The exact parameters of our networks are covered in the **todo: Appendix**.

3.2 Generative Models for Point Clouds

Raw point cloud GAN (r-GAN). The first version of our generative model operates directly on the raw 2048×3 point set input – to the best of our knowledge this work is the first to present a GAN for point clouds. The architecture of the discriminator is identical to the AE (modulo the filter-sizes and the number of neurons), without any batch-norm and with leaky ReLUs Maas et al. (2013) instead or ReLUs. The output of the last fully connected layer is fed into a sigmoid neuron. The generator takes as input a 128-dimensional noise vector and maps it to a 2048×3 output by 5 FC-ReLU layers.

Latent-space GAN (l-GAN). In our latent-space GAN, instead of operating on the raw point cloud



input, we pass the data through our pre-trained autoencoder, trained separately for each object class with the EMD (or Chamfer) loss function. Both the generator and the discriminator of the GAN then operate on the 128-dimensional bottleneck variable of the AE. Finally, once the GAN training is over, the output of the generator is decoded to a point cloud via the AE decoder. The architecture for the l-GAN is significantly simpler than the one of the r-GAN. We found that very shallow designs for both the generator and discriminator (in our case, one hidden FC layer for the generator and two FC for the discriminator) are sufficient to produce realistic results.

Gaussian Mixture Model. In addition to the l-GANs, we also train a family of Gaussian Mixture Models (GMMs) on the latent spaces learned by our AEs. We fitted GMMs with varying numbers of Gaussian components, and experimented with both diagonal and full covariance matrices for the Gaussians. The GMMs can be turned into point-cloud generators by first sampling the latent-space from the GMM distribution and accordingly using the AE's decoder, similarly to the l-GANs.

4 Evaluation and Results

Our source for shapes is the ShapeNet repository (Chang et al., 2015); we pre-center all shapes into a sphere of diameter 1. Unless otherwise stated, we train specific per-class models, and split the models in each class into training/validation/testing set using a 85%-10%-5% split.

4.1 Evaluating the latent representation

Classification. A common technique for evaluating the quality of unsupervised representation learning algorithms is to apply them as a feature extractor on supervised datasets and evaluate the performance of linear models fitted on top of these features. We use this technique to evaluate the performance of the latent "features" computed by our AE. For this experiment to be meaningful, the AE was trained across all different shape categories: we used 57,000 models from ShapeNet from 55 categories of man-made objects. Exclusively for this experiment, we used a bigger bottleneck of 512, increased the number of neurons and applied batch-norm to the decoder as well. To obtain features for an input 3D shape, we feed forward to the network its point-cloud and extract the 512-dimensional bottleneck layer vector. This feature is then processed by a linear classification SVM trained on the de-facto 3D classification benchmark of ModelNet Wu et al. (2015). Table 1 shows comparative results. Note that previous state of the art Wu et al. (2016) uses several layers of a GAN to derive a 7168-long feature; our 512-dimensional feature is more intuitive and parsimonious.

todo: add manually references (see commented)

Dataset	SPH[]	LFD[]	T-L-Net[]	VConv-DAE[]	3D-GAN[]	ours, (EMD)	ours,(CD)
MN10	79.8%	79.9%	-	80.5%	91.0%	**95.3**%	95.0%
MN40	68.2%	75.5%	74.4%	75.5%	83.3%	84.0%	**84.7**%

Table 1: Classification performance on ModelNet40 and ModelNet10. All methods train a linear SVM with features derived in an unsupervised manner.

The decoupling of latent representation from generation allows flexibly choosing the AE loss, which can improve the learned feature. On ModelNet10, which includes primarily larger objects than ModelNet40, an AE trained with the EMD loss performs slightly better than one trained with the CD. On the other hand, when the variation within the collection increases, CD produces better results. This is perhaps due to its more local and less smooth nature, which allows it to understand rough edges ands some high frequency geometric details. Finally, note that since our AEs were not trained on ModelNet, this experiment also demonstrates the domain-robustness of our learned features.

Qualitative Evaluation. To visually assess the quality of the learned representation, we show some reconstruction results in Fig. 1. Here, we use our AEs to encode samples from the *test* split of the ground truth dataset (the leftmost of each pair of images) and then decode them and compare them visually to the input (the rightmost image). These results shows the ability of our

learned representation to *generalize* to unseen shapes. In addition to reconstruction, our learned latent representation enables a number of interesting shape editing applications, including shape interpolations, part editing and shape analogies. We showcase some results in the appendix.

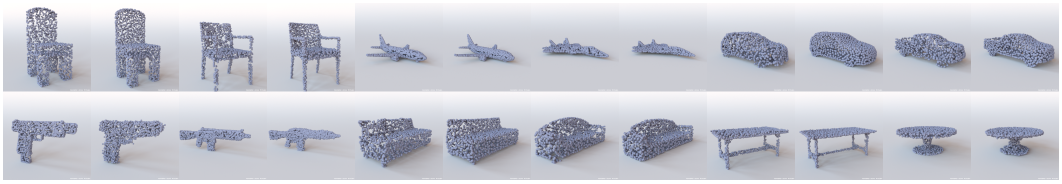


Figure 1: Reconstructions of unseen shapes from the *test* split of the input data. The leftmost image of each pair shows the ground truth shape, the rightmost the shape produced after encoding and decoding using our class-specific AEs.

4.2 EVALUATING THE GENERATIVE MODELS

We train and compare a total of five generative models on the data distribution of point-clouds of the *chair* category. We start from the two AEs with a 128-dimensional bottleneck, trained with the CD or EMD loss respectively – referred to as AE-CD and AE-EMD. Both AEs were stopped at the epoch at which the average reconstruction error with respect to our validation dataset was minimized (see Appendix, Fig. 11). We train an l-GAN in each of the AE-CD and AE-EMD latent spaces. In the space associated *only* with the AE-EMD we train a further two models: an identical (architecture-wise) l-GAN that utilizes the Wasserstein objective with gradient-penalty (Gulrajani et al., 2017), and a family of GMMs. Lastly, we also train an r-GAN directly on the point cloud data.

Model Selection. 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 (Section 2). 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 epochs at which the various models were selected using the JSD criterion are shown in Table 3. Using the same criterion, we also select the number and covariance type of Gaussian components for the GMM, and obtain the optimal value of 32 components. GMMs performed much better with full (not diagonal) covariance matrices, suggesting strong correlations between the latent dimensions; see appendix (Fig. 14). When using MMD-CD as the selection criterion, we obtain models of similar quality and at similar stopping epochs (appendix, Table 7); the optimal number of Gaussians in this case was 40.

Metric	r-GAN	Wu[**todo:**]	l-GAN (CD)	l-GAN (EMD)	l-WGAN (EMD)	GMM (EMD)
JSD	0.1660	0.1705	0.0372	0.0188	0.0077	**0.0048**
Classification	84.10	87.00	96.10	94.53	89.35	87.40
MMD-CD	0.0017	0.0042				

Table 2: Evaluating 5 generators on train-split of *chair* dataset on epochs/models selected via minimal JSD on the validation-split. Note that the average classification score attained by the ground-truth point clouds was 84.7%.

Quantitative Evaluation. Upon selection of the models, we compare them with respect to their capacity to generate synthetic samples. In two different sets of experiments, measure how well the distribution of the generated samples resembles both the train and test splits of the ground truth distribution, by using our models to generate a set of synthetic point clouds and using the metrics from Section 2 to compare against the train or test set distributions respectively. The train split results are reported in Table 2. We also show the average classification probability for those samples being recognized as a chair using the PointNet classifier Qi et al. (2016b), which is state of the art for

Method	Epoch	JSD	MMD-CD	MMD-EMD	COV-EMD	COV-CD
Raw-GAN	1700	0.1764	0.0020	0.1230	19.0	52.3
Latent-GAN (Chamfer)	300	0.0486	0.0020	0.0796	32.2	59.4
Latent-GAN (EMD)	100	0.0308	0.0023	0.0697	57.1	59.3
Latent-WGAN (EMD)	1800	0.0227	0.0019	0.0660	66.9	67.6
GMM-Full-32	-	**0.0202**	**0.0018**	**0.0651**	**67.4**	**68.9**

Table 3: Evaluating 5 generators on test-split of *chair* dataset on epochs/models selected via minimal JSD on the validation-split. The reported scores are averages of 3 pseudo-random repetitions.**todo: check caption**

classifying point clouds. A similar experiment is ran to measure how well the synthetic samples match the test split dataset; here, we repeat the experiment with three pseudo-random seeds and report the average measurements in Table 3, for various comparison metrics. Perhaps surprisingly, training a simple Gaussian mixture model in the latent space of the EMD-based AE yields the best results in terms of both fidelity and coverage. Furthermore, GMMs are particularly easy to train. Additionally, the achieved fidelity and coverage are very close to the reconstruction baseline - for example, the AE-EMD achieved an MMD-EMD of 0.05 with respect to the ground truth training data, which is comparable with the MMD-EMD value of 0.06 achieved by the GMMs with respect to the test data. See the appendix for more on the reconstruction baseline.

Note: The number of synthetic point clouds we generate for the train split experiment is equal to the size of the train dataset. For the test split experiment, as well as for the validation split comparisons done for model selection, we generate synthetic datasets that are three times bigger than the ground truth dataset (the test resp. validation set); this is possible due to the relatively small size of the test resp. validation sets, and helps reduce sampling bias. This is only necessary when measuring MMD or Coverage statistics.

olga: "train and test are the same" - update - is this meant to be shown by the JSD? I'm not sure we can claim it.**todo: Panos is making MMD-CD for Table 2 and hopefully this will show this. Also refer to Fig. 12**

Fidelity of Metrics. In Table 3 we note that the MMD-CD distance to the test set appears smaller for the r-GANs. This seeming advantage of the r-GANs is counter to what a qualitative inspection of the results yields. This effect is actually due to a bias panos: I think i prefer something like "short-coming" present in the CD metric, which we demonstrate by Fig. 2: here, we used an r-GAN and two l-GANS (with CD-AE and EMD-AE) to generate a synthetic set of point clouds of size 3x the size of the test dataset. We choose a ground truth point cloud from the test set (the leftmost image of each triplet), and find its nearest neighbor in the synthetic set under the CD; we measure the difference between this nearest neighbor and the ground truth point set using both CD and EMD. Note that the CD values miss the fact that the r-GAN results are obviously of lesser quality. The underlying reason appears to be that r-GANs tend to generate point clouds with most points concentrated with high density in specific areas in space, where it is more common for the shape to have a presence (e.g. the seat of chairs in the figure). This means that one of the two terms in the CD –namely, the one going from the synthetic point cloud to the ground truth– is likely to be very small for r-GAN results, which indicates a bias. olga: check check check The "blindness" of the CD metric to partial matches between shapes has the interesting side-effect that the CD-based coverage metric is consistently bigger than that reported by EMD, as noted in Table 3. Similar observations were made when measuring the reconstruction quality of the AEs themselves using the CD distance - see Appendix for more details. Instead, the EMD distance promotes a one-to-one mapping and thus correlates more strongly to visual quality; this means that it heavily penalizes the r-GAN result both in terms of MMD and coverage.

Training trends. yianni: we need to boost this shitFig. 4 shows the behaviour of the models during training. On the left, we plot the JSD distance between the ground truth test set and synthetic datasets generated by the various models at various epochs of training. On the right, we also plot the MD-EMD together with the Coverage-EMD distances between the same two sets, where larger marker symbols denote a higher epoch. In general, r-GAN struggles to provide good coverage of the

test set no matter the metric used; which alludes to the well-established fact that end-to-end GANs are generally difficult to train. The l-GAN (CD AE) performs better in terms of fidelity with much fewer epochs as measured by JSD/MMD-EMD, but its coverage remains low. We attribute this to the CD promoting unnatural topologies where different point clouds are clumped together olga: what does this mean (also cf. Fig. 2 that visually shows this phenomenon. Switching to an EMD-based AE for the representation and otherwise using the same latent GAN architecture (l-GAN, EMD AE), yields a dramatic improvement in coverage and fidelity. Both l-GANs though suffer from the known issue of mode collapse: Half-way through training, first coverage starts dropping with fidelity still at good levels, which implies that they are overfitting a small subset of the data. Later on, this is followed by a more comprehensive collapse, with coverage dropping as low as 0.5%. Switching to a latent WGAN largely eliminates this collapse, as expected.

Comparisons to voxel-based methods. To the best of our knowledge we are the first to propose GANs on point-cloud data. To find out how our models fare against other 3D generative methods, in Table 2 and Fig. 4 we compare to a recent voxel-grid based approach (Wu et al., 2016) in terms of the JSD on the training set of the *chair* category - other shape categories can be found in the appendix. We convert their voxel grid output into a point-set with 2048 points by performing farthest-point-sampling on the isosurface of the grid values. Per the authors' suggestion, we used an isovalue parameter of 0.1 and isolated the largest connected component from the isosurface. The r-GAN mildly outperforms Wu et al. (2016) in terms of its diversity (as measured by olga: pending: MMD), while also creating realistic-looking results, as shown by the classification score. The l-GAN performs even better, both in terms of classification and diversity, with less training epochs. Note also that the training time for one epoch of the l-GAN is more than an order of magnitude smaller than for the r-GAN, due to its much smaller architecture and dimensionality. For fairness, since Wu et al. (2016) operates on voxel grids, it is not necessarily on equal standing when it comes to generating point clouds.

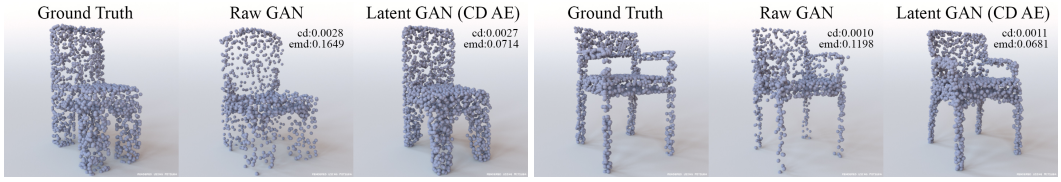


Figure 2: The CD distance is less faithful than EMD to visual quality of synthetic results. In particular, the CD is positively biased towards r-GAN results, due to the presence of high-density areas in the synthesized point sets. See text for more details.

Qualitative Evaluation. In Fig. 3, we show some synthetic results produced by our l-GANs (top row) and the 32-component GMM, both trained on the EMD-AE latent space. We notice high quality results from either model - this highlights the strength of our learned representation, which makes it possible for the simple GMM model to perform well. The shapes (after decoding) corresponding to the 32 means of the Gaussian components can be found in the appendix (Fig. 15), as well as results using the r-GAN (Fig. 10). The l-GAN produces crisper and less noisy results than the r-GAN, demonstrating an advantage of using a good structural loss on the decoupled, pre-trained AE.

Extensions to multiple classes yianni: since people "kind of expect" that multi-class is doable with W-gans how can we phraze this paragraph more gently? We have performed preliminary experiments

Figure 3: Synthetic point clouds generated by samples produced with l-GAN (top) and 32-component GMM (bottom), both trained on the latent space of an AE using the EMD loss. **todo: change text in image**

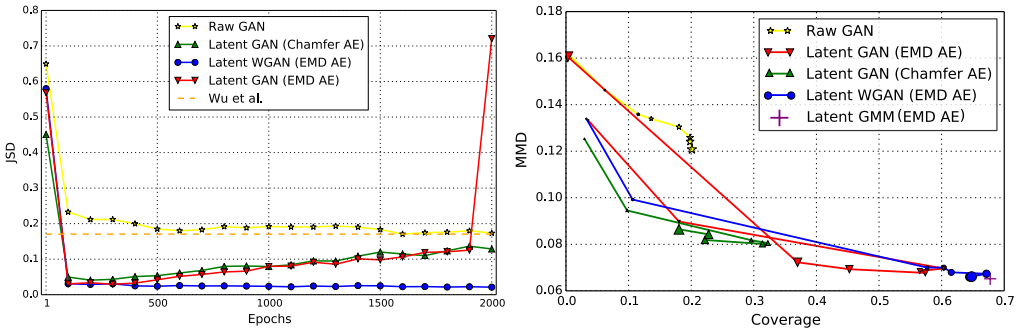


Figure 4: Training trends for the various generative models, in terms of coverage of /fidelity to the ground truth test dataset. On the right, the curve markers indicate epochs 1, 10, 100, 200, 400, 1000, 1500, 2000, with larger symbols denoting higher epochs. See text for more details.

with an EMD-AE trained on a mixed set containing point clouds from 5 categories (*chair, airplane, car, table, sofa*). The training and testing datasets for this AE were constructed by randomly picking and adding models from each class; 2K models per class for the training set, 200 models for testing and 100 for validation. The multi-class AE has the same bottleneck size of 128 and was trained for 1000 epochs. We compare against the class-specific AEs with the 85-5-10 train-val-test-split, which we trained for 500 epochs. The precise AE model in all cases was selected based on the minimal reconstruction loss on the the respective validation set. On top of all six AEs, we train six l-WGANs for 2K epochs, and evaluate their fidelity/coverage using the MMD-CD between the respective testing sets and a synthesized dataset of 3x the size, as above. It turns out that the l-WGANs based on the multi-class AE perform similarly to the dedicated class-specifically trained ones (Table 4). A qualitative comparison (Fig. 5) also reveals that by using a multi-class AE-EMD we do not sacrifice much in terms of visual quality compared to the dedicated AEs.

Figure 5: Synthetic point clouds generated by samples produced with l-WGANs trained in the latent space of an AE-EMD trained on a multi-class dataset.

	airplane	*car*	*chair*	*sofa*	*table*	average	multi-class
train	0.0004	0.0006	0.0015	0.0011	0.0013	**0.0010**	0.0011
test	0.0006	0.0007	0.0019	0.0014	0.0017	**0.0013**	0.0014

Table 4: MMD-CD measurements for l-WGANs stopped at the two-thousand epoch and trained on the latent spaces of dedicated (left 5 columns) and multi-class EMD-AEs (right column) .

Limitations. Fig. 6 shows some failure cases of our models. Chairs with rare geometries (left two images) are sometimes not faithfully decoded. Additionally, Middle failure case: The AEs can loose high-frequency geometric details (lack of back whole) and alter the style of their input. **todo: here** Right failure case: while the r-Gan creaes chairs that are visually easily recognizable as chairs, the same architecture has a harder time on cars: i.e., there is a ample opportunity for more robust raw-point cloud GANs.

5 Related work

Other 3D work

Variational Autoencoders (VAE) Training Gaussian mixture models (GMM) in the latent space of an autoencoder is closely related to VAEs (Kingma & Welling, 2013). One documented issue

Figure 6: **todo: Failure Cases**

with VAEs is over-regularization: the regularization term associated with the prior, is often so strong that reconstruction quality suffers (Bowman et al., 2015; Sønderby et al., 2016; Kingma et al., 2016; Dilokthanakul et al., 2016) The literature contains methods that start only with a reconstruction penalty and slowly increase the weight of the regularizer. In our case, we find that fixing the AE before we train our generative models yields good results.

6 TODO's

olga: All the TODOs go here

todo: yianni: Mention code is on github, on the intro?

todo: Secondary: Add error-bars on Chamfer-EMD AE-bottleneck reconstuction curves

todo: add bf to argmax-min of the tables

todo: decide if we will add interpolations on main

References

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *NIPS 2016 Workshop on Adversarial Training. In review for ICLR*, volume 2016, 2017.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3837–3845, 2016.

Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pp. 1486–1494, 2015.

Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. *CoRR*, abs/1612.00603, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014.

Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *CoRR*, 2015.

Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3d shape segmentation with projective convolutional networks. *CoRR*, abs/1612.02808, 2016.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P Kingma, Tim Salimans, and Max Welling. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, 2016.

Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30 th International Conference on Ma- chine Learning (ICML-13)*, 2013.

Anh Nguyen, Jason Yosinski, Yoshua Bengio, Alexey Dosovitskiy, and Jeff Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv preprint arXiv:1612.00005*, 2016.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016a.

Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016b.

Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 5648–5656, 2016c. doi: 10.1109/CVPR.2016.609.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, (2):99–121, 2000. ISSN 1573-1405. doi: 10.1023/A:1026543900054.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

Raif M. Rustamov, Maks Ovsjanikov, Omri Azencot, Mirela Ben-Chen, Frédéric Chazal, and Leonidas Guibas. Map-based exploration of intrinsic shape differences and variability. *ACM Trans. Graph.*, 32(4):72:1–72:12, July 2013. ISSN 0730-0301. doi: 10.1145/2461912.2461959. URL `http://doi.acm.org/10.1145/2461912.2461959`.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2226–2234, 2016.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. How to train deep variational autoencoders and probabilistic ladder networks. *arXiv preprint arXiv:1602.02282*, 2016.

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 945–953, 2015. doi: 10.1109/ICCV.2015.114.

Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. Dense human body correspondences using convolutional networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 1544–1553, 2016. doi: 10.1109/CVPR.2016.171.

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 D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 82–90. Curran Associates, Inc., 2016.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 1912–1920, 2015. doi: 10.1109/CVPR.2015.7298801.

Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas J. Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Trans. Graph.*, (6):210:1–210:12, 2016a.

Li Yi, Hao Su, Xingwen Guo, and Leonidas J. Guibas. Syncspeccnn: Synchronized spectral CNN for 3d shape segmentation. *CoRR*, 2016b.

A AE RECONSTRUCTION QUALITY

Table 5 shows the reconstruction quality of the two AEs (CD- and EMD-based), in terms of the JSD of the reconstructed datasets (train or test, as noted in the table) with respect to their ground truth counterparts. Note that, in general, the quality of reconstruction is consistent between the training and test datasets, indicating that the AEs are indeed able to generalize. Also note the preference of the MMD-CD metric for the AE-CD latent representation, which may be another instance of the CD bias as outlined in Section 4.2.

Method	JSD (Tr)	JSD (Te)	MMD-CD (Tr)	MMD-EMD (Tr)
Chamfer AE	0.0216	0.0243	**0.0004**	0.0753
EMD AE	**0.0028**	**0.0067**	0.0005	**0.0527**

Table 5: Effect of loss-type for AE reconstructions. The EMD loss gives rise to reconstructions with significantly better JSD compared to Chamfer. MMD-measurements favor the AE that was trained with the same loss under which the MMD measurement is computed. (Tr: Train split, Te: Test split)

B APPLICATIONS OF THE LATENT SPACE REPRESENTATION

For shape editing applications, we use the embedding we learned with the EMD-AE 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. (2016a) 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,

Figure 7: Editing parts in point clouds using vector arithmetic on the AE latent space. Left to right: tuning the appearance of cars towards the shape of convertibles, adding armrests to chairs, removing handle from mug.

Figure 8: Interpolating between different point clouds, using our latent space representation. Note the interpolation between structurally and topologically different shapes.

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 Figure 7. Note that the volume of chairs with armrests is on average 10% larger than the chairs without, which is reflected in the output of this process.

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 (Figure 8). 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 types, cfg. the second row for an interpolation between a bench and a sofa. More results are available in the additional material.

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. 9 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).

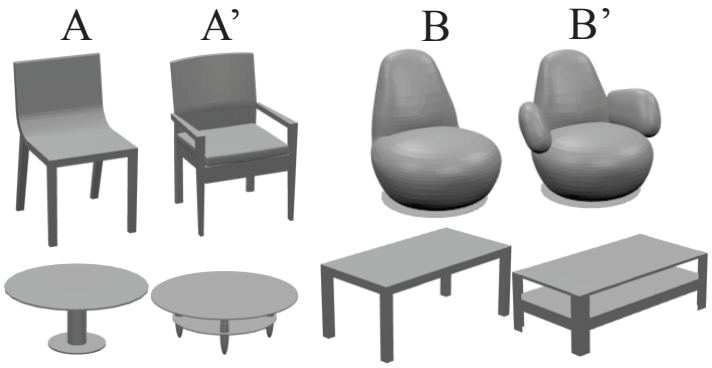


Figure 9: **todo: Shape Analogies using our learned representation. Shape B' relates to B in the same way that shape A' relates to A.**

Figure 10: **todo: Generative results with R-GAN.** olga: The car was the single mode within 30 samples - I can search more if needed.

C Further Evaluation and Results

olga: make sure all of this is cited somewherer in the main text.

D AE Details

With the exception of the AE used for **todo: ref the one with the 512 bneck** the first 4 MLP layers of our AE-encoder were implemented as 1D-convolutions with ReLUs and stride of 1, i.e. treating each point independently. Their filter sizes were 64, 128, 128, 256 and k respectively, with k being the bottle-neck size. The decoder had 3 FC-ReLU layers with 256, 256, 2048 $\times$ 3 neurons each. For training we used ADAM with initial learning rate of 0.0005, beta-1 of 0.9 and a batch size of 50 point-clouds.

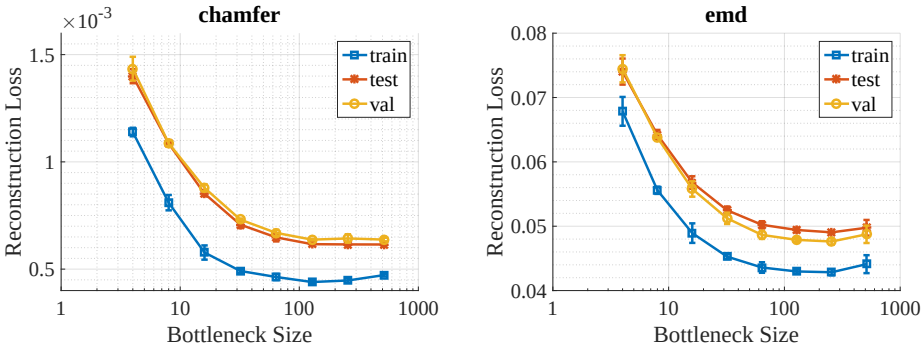


Figure 11: The optimal bottleneck size was fixed at 128 by observing the reconstruction loss of the AEs, shown here for various bottleneck sizes.

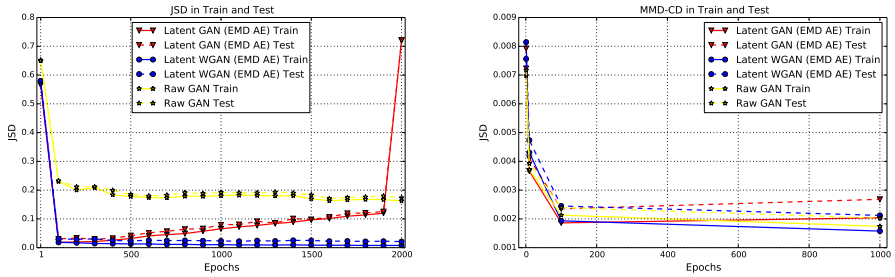


Figure 12: **todo: Generalization error of metrics. JSD (Left) and MMD-CD (Right) values of various GANs with regards to the training and testing distributions, at various training epochs. For this MMD-CD the synthetic set sizes were identical to those of their comparators, train/test respectively.** olga: train and test are the same.

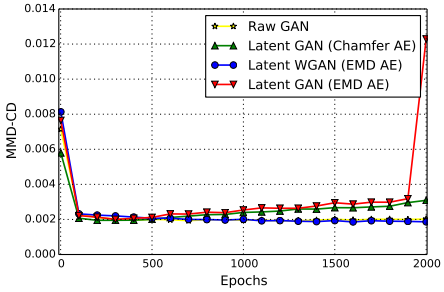


Figure 13: **todo: Training trends in terms of the MMD-CD metric (syntetic samples were boosted by 3). Compare this to Fig. 4, left, and note the similar behaviour.**

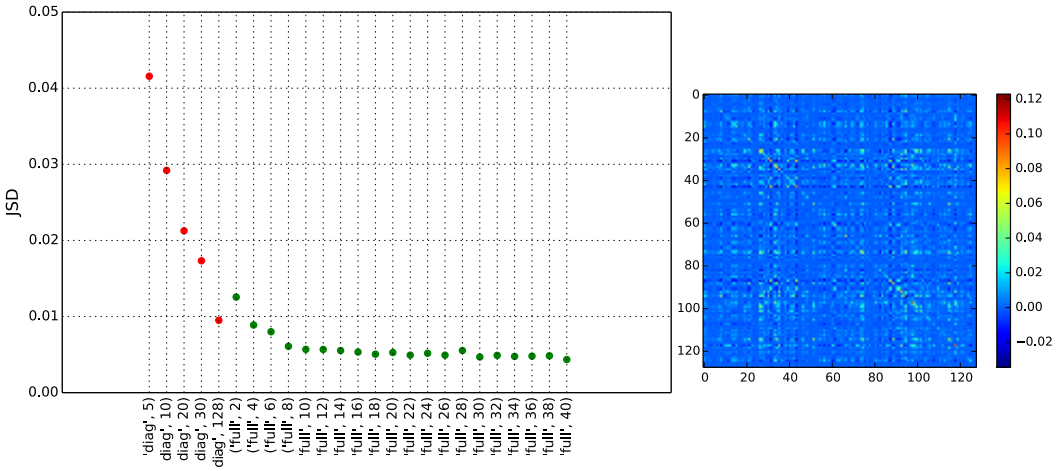


Figure 14: GMM model selection.GMMs with a varying number of gaussians and covariance type are trained on the latent space learned by and AE trained with EMD and a bottleneck of 128. Models with a full covariance matrix achieve significantly smaller JSD than models trained with diagonal covariance. For those with full covariance, 30 or more clusters seem sufficient to achieve minimal JSD. olgaRight: a typical covariance matrix blah blah. olga: replace the covariance image with average of absolute values.

Class	Wu et al. (2016)	L-GAN (AE EMD)		Full GMM/32 (AE EMD)	
		train	test	train	test
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: **todo: JSD Comparison between Wu et al. (2016) and our generative models. Full GMM/32 stands for... Note that the l-GAN here uses the same "vanilla" adversarial objective as Wu et al. (2016).**

Method	Epoch	JSD	MMD-CD	MMD-EMD	COV-EMD	COV-CD
Raw-GAN	1350	0.1893	0.0020	0.1265	19.4	54.7
Latent-GAN (Chamfer)	300	0.0463	0.0020	0.0800	32.6	58.2
Latent-GAN (EMD)	200	0.0319	0.0022	0.0684	57.6	58.7
Latent-WGAN (EMD)	1700	0.0240	0.0020	0.0664	64.2	64.7
GMM-Full-40	-	**0.0182**	**0.0018**	**0.0646**	**68.6**	**69.3**

Table 7: **todo: Evaluation of five generators on test-split of *chair* data on epochs/models that were selected via minimal MMD-CD on the validation-split. The reported scores are averages of three pseudo-random repetitions. Compare this with Table 3. Note that the overall quality of the selected models remains the same, irrespective of the metric used for the selection.**

E R-GAN DETAILS

The discriminator's first 5 layers are MLPs with filter sizes of $\{64, 128, 256, 256, 512\}$ each and leaky-ReLU between them. They are followed by a feature-wise max-pool. The last 2 FC-leaky-ReLU layers have $\{128, 64\}$, neurons each leading to single sigmoid neuron. We use a leak of 0.2 units.

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 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.

F 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 $\{128, 128\}$ neurons each. When used the l-Wasserstein-GAN, we used a gradient penalty regularizer $\lambda = 10$. The training parameters (learning rate, batch size) and the generator's noise distribution were the same used for the r-GAN.

Figure 15: The 32 centers of the GMM fitted to the latent codes, and decoded using the decoder of the AE-EMD.