DeepEMD: A Transformer-based Fast Estimation of the Earth Mover's Distance

Atul Kumar Sinha [1] **François Fleuret** [1]

Abstract

The Earth Mover's Distance (EMD) is the measure of choice to assess similarity between point clouds. However the computational cost of standard algorithms to compute it makes it prohibitive as a training loss, and the standard approach is to use a surrogate such as the Chamfer distance. We propose instead to use a deep model dubbed DeepEMD to directly get an estimate of the EMD. We formulate casting the prediction of the bipartite matching as that of an attention matrix, from which we get an accurate estimate of both the EMD, and its gradient. Experiments demonstrate not only the accuracy of this model, in particular even when test and train data are from different origins. Moreover, in our experiments, the model performs accurately when processing point clouds which are several times larger than those seen during training. Computation-wise, while the complexity of the exact Hungarian algorithm is $O(N^3)$, DeepEMD scales as $O(N^2)$, where N is the total number of points. This leads to a $100\times$ wall-clock speed-up with 1024 points. DeepEMD also achieves better performance than the standard Sinkhorn algorithm, with about $40\times$ speed-up. The availability of gradients allows DeepEMD to be used for training a VAE, leading to a model with lower reconstruction EMD than a standard baseline trained with Chamfer distance.

1. Introduction

The *earth mover's distance* (EMD), also known as *Wasserstein distance* is a distance between distributions that is defined as the minimum total of mass-time-distance displacement needed to transform one distribution to the other. In the case of uniform distributions over a finite number of points, it turns into a distance between point clouds that corresponds to finding the one-to-one matching that minimizes the sum of the distances between pairs of matched points. Since there is no inherent ordering in point cloud data, computing the EMD between two point clouds involves finding a matching based on the euclidean distance between points. The matching is constrained to be bipartite so that one point cloud is completely transformed to the other, without any fractional assignment, and the transport cost is minimal. EMD reflects the notion of nearness properly, does not have quantization/binning and non-overlapping support problems of most other metrics, e.g., f-divergences, total variation distance, etc.

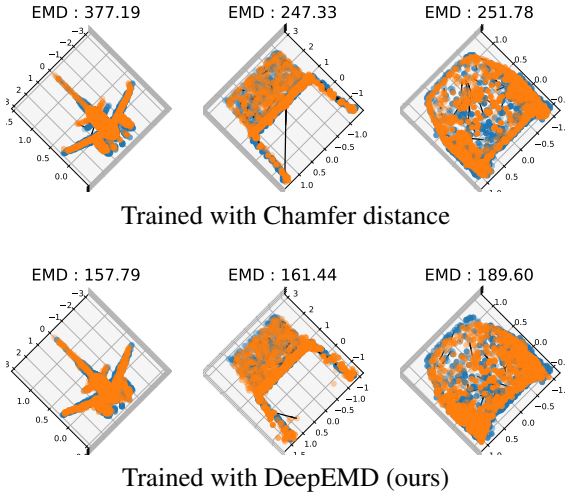


Figure 1. Example point clouds (blue) and their VAE reconstructions (orange) when trained with different reconstruction losses. Training with DeepEMD (bottom) consistently achieves lower reconstruction error (EMD, shown on top of each example) than with the standard Chamfer distance (top).

Consider two point clouds $X = \{x_i\}_{i=1}^N$ and $Y = \{y_j\}_{j=1}^N$, where $x_i, y_j \in \mathbb{R}^d$. The EMD between the two point clouds can be computed as,

$$\mathrm{EMD}(X,Y) = \min_{\phi \in \mathcal{M}(X,Y)} \sum_{x \in X} \|x - \phi(x)\|_2, \qquad (1)$$

where $\mathcal{M}(X,Y)$ is the set of 1-to-1 (bipartite) mappings from X to Y. In addition to the distance, the optimal match-

[1]University of Geneva, Geneva, Switzerland. Correspondence to: Atul Kumar Sinha <atul.sinha@unige.ch>, François Fleuret <francois.fleuret@unige.ch>.

Presented at the 2^{nd} Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML) at the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. 2023.

ing ϕ^* is also interesting for some applications. Since directly optimizing EMD is computationally expensive, most methods in the literature rely on Chamfer Distance (CD) as a proxy similarity measure or reconstruction loss. CD can be computed as,

$$\mathrm{CD}(X,Y) = \sum_{x\in X} \min_{y\in Y} \|x-y\|_2^2 + \sum_{y\in Y} \min_{x\in X} \|x-y\|_2^2,$$

and in $O(N^2)$ time complexity. The CD solution leads to a non-bipartite one-to-many matching between $x \to y$ and vice versa. We can also use the L_2 measure with $d = \|x - y\|_2$ instead $d = \|x-y\|_2^2$ to make it comparable to EMD. of Note that the above EMD for point clouds is related to the Wasserstein-2 metric (see appendix § B for details). The utility of EMD is limited by the $O(N^3)$ computational cost of evaluating it. There have been several research efforts to circumvent this issue in various application settings.

This is the case for application to point clouds where N is usually in the range of several thousands. Kim et al. (2021) trains a variational auto-encoder with CD as the reconstruction loss. EMD is still the metric of choice for evaluating point cloud generative models (Huang et al., 2022; Luo & Hu, 2021; Kim et al., 2021; Yang et al., 2019; Shu et al., 2019; Achlioptas et al., 2018). Another issue is disparity between performance measures (minimum matching distance, coverage, etc.) computed with EMD and CD, the comparisons are contradictory and often inconsistent across measures. CD is usually insensitive to mismatched local density while EMD is dominated by global distribution and overlooks the fidelity of detailed structures (Wu et al., 2021). Wu et al. (2021) proposes a new similarity metric called Density-aware Chamfer distance (DCD) to tackle these issues. DCD is derived from CD and can also be computed in $O(N^2)$ time complexity. Urbach et al. (2020) proposed Deep Point Cloud Distance (DPDist) which measures the distance between the points in one cloud and the estimated continuous surface from which the other point cloud is sampled. The surface is estimated locally by a network using the 3D modified Fisher vector representation.

In the optimal transport literature, several efforts have been taken towards improving the statistical and computational properties. Recently, Chuang et al. (2022) proposed Information Maximizing Optimal Transport (InfoOT) which is an information-theoretic extension of optimal transport based on kernel density estimation of the mutual information (MI) which introduces global structure into OT maps. The resulting solution maximizes the MI between domains while minimizing geometric distance and improves the capability for handling data clusters and outliers. Other approaches focus on regularizing the OT problem for making it smooth and strictly convex (Cuturi, 2013; Flamary et al., 2016; Genevay et al., 2018; Blondel et al., 2018). *Sinkhorn distances* (Cuturi, 2013) smooth the classic OT problem with an entropic regularization term and can be computed through Sinkhorn's matrix scaling algorithm at a speed that is several orders of magnitude faster than that of transport solvers. We provide more details in the appendix.

2. Method

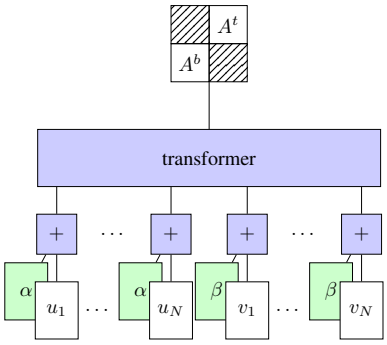


Figure 2. The transformer model we use for DeepEMD predicts directly the bipartite graph as an attention matrix.

We are interested in building a model which operates on a pair of point clouds (U, V) as input, where $U, V \in \mathcal{D}^N$, $U = \{u_i\}_{i=1}^N$, $V = \{v_j\}_{j=1}^N$, $u_i, v_j \in \mathbb{R}^D$, and N is the cardinality of the point clouds. We denote the earth mover's distance between them as $d = \mathrm{EMD}(U, V)$ where $d \in \mathbb{R}$. The goal of the model is to predict d and ∇d. Also, let $M \in \{0,1\}^{N\times N}$ denote the ground truth bipartite matching from EMD, where $M_{i,j} = 1$ indicates that u_i is matched to v_j and vice-versa. Bipartite-ness implies $\forall j, \sum_i M_{i,j} = 1$ and $\forall i, \sum_j M_{i,j} = 1$. Since point clouds are unordered and invariant to elementwise permutation, we seek mappings $f : \mathcal{D}^N \times \mathcal{D}^N \to \mathbb{R}$ which are permutation invariant for any permutations π and π', i.e.,

$$f(U, V) = f(\pi(U), \pi'(V)),$$

We propose *DeepEMD* composed of a sequence of multi-head full attention layers, followed by a prediction head which is also a full attention layer, but with a single head. We concatenate learned cloud-specific positional embedding to indicate if a point originates from U or V, we concatenate the points sequence and feed the resulting $I = U \cup V$ as input to the model. The group-id embedding helps the model in modulating attention locally within a point cloud as well as globally across both point clouds. We tried other variants with self-attention layers, cross attention layers, and an alternating mixture of both, but found full attention over both point clouds to be best performing.

For our problem, we get the input X for the transformer by adding positional embeddings to I. Let $\vec{0}^n$ and $\vec{1}^n$ denote a vector of n zeros and ones, respectively. X is obtained as,

$$P = \vec{0}^n \cup \vec{1}^n, \quad X = I + W^P[P] \quad \text{(indexing)}$$

The intermediate feature $t(X)$ from the transformer encoder (see appendix for details) has the same number of elements as X with each element now being a contextualized representation for the corresponding point in the input. Further, these intermediate representation are fed into a single-head attention layer which outputs the attention matrix as,

$$K = t(X)W^K, \quad Q = t(X)W^Q, \quad A = \frac{QK^\top}{d_k}$$
$$A^t = A_{:n,n:} \quad A^b = A_{n:,:n}$$

Here, A is a $2N \times 2N$ matrix and we slice the top-right block (first N rows and last N columns) as A^t and bottom-left block (last N rows and first N columns) as A^b. $A^t_{i,j}$ can be interpreted as the relatedness of u_i with v_j. Similarly, $A^b_{i,j}$ can be interpreted as the relatedness of v_i with u_j. We define the loss as the average of the cross-entropies as,

$$l(U,V) = \frac{1}{N}\sum_{i=1}^{N} \text{CE}(A^t_{i,\cdot}, M_{i,\cdot}) + \frac{1}{N}\sum_{i=1}^{N} \text{CE}(A^b_{i,\cdot}, M_{\cdot,i})$$

The EMD is then estimated with the predicted matching as,

$$\phi^b(i) = \underset{j}{\text{argmax}}\, A^b_{i,j}, \quad \phi^t(i) = \underset{j}{\text{argmax}}\, A^t_{i,j}$$
$$\hat{d} = \frac{1}{2}\left(\sum_i \|u_i - v_{\phi^t(i)}\| + \sum_i \|v_i - u_{\phi^b(i)}\|\right)$$

We also compare with a MLP baseline detailed in the Appendix § D.

3. Experiments

3.1. Datasets

We consider three different datasets for our experiments - *Syn2D*, ShapeNet (Chang et al., 2015) and ModelNet40 (Wu et al., 2015). *Syn2D* consists of 2D point clouds generated by sampling points on squares and circles (see Fig. 7). ShapeNet and ModelNet40 are datasets of 3D point clouds derived from 3D CAD models for different real world objects like chair, car, airplane, etc. In order to improve and assess generalization, we augment train and test splits with synthetic perturbations. We provide more details about the datasets and these augmentations in Appendix § C.

3.2. Performance Measures

We consider various measures to assess performance of EMD approximation methods, for both distance and matching estimation. We compare accuracy and computation time to that of Sinkhorn and CD (see § 1). We summarize the measures considered in Appendix § A.1.

3.3. Results

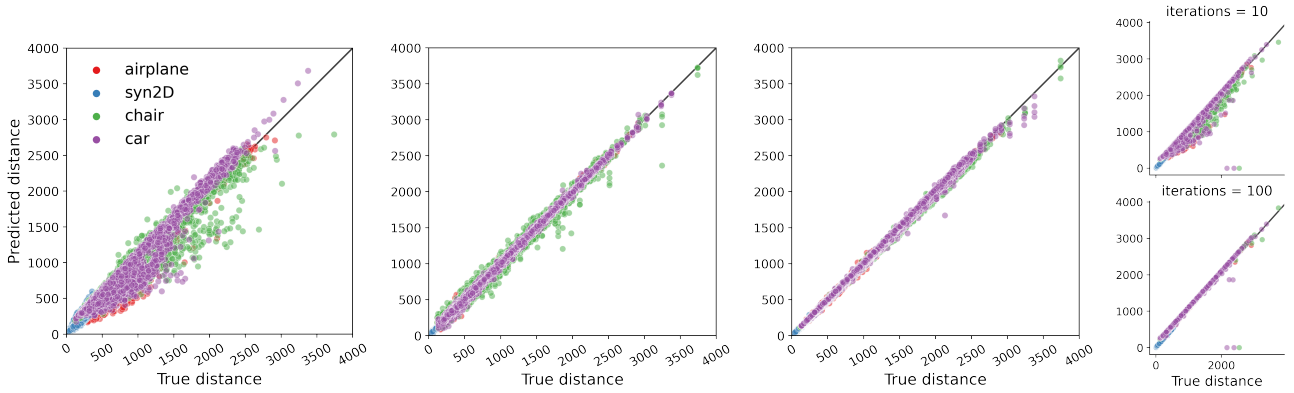
EMD Prediction. Fig. 3 shows the scatter plot of the true EMD vs. approximate EMD predicted from our trained models on the validation split for *Syn2D* and ShapeNet datasets. Note that the validation split also contains augmentations as discussed in Sec. 3.1. We also validate on specific splits and the results are shown in the appendix. The plots indicate that both DeepEMD (Fig. 3c) and MLP baseline (Fig. 3b) approximate the EMD faithfully.

Matching/Gradient Prediction. Estimating the matching and gradient of the distance is particularly important for training models with DeepEMD as a surrogate. Note that gradient of a point from true EMD is always along the matched point in the other point cloud. Fig. 4a shows cdf of cosine similarity between the true and estimated gradient for all points across all point clouds collected together for *Syn2D*, while Figs. 4b, 4c, and 4d for ShapeNet.

Out-of-distribution generalization. The generalisation of the prediction to a novel distribution is particularly important for a surrogate metric. We test the out-of-distribution behaviour of our models in two different settings : Table 1 shows the generalization performance of the model trained on a single category of ShapeNet and tested on validation split of multi-category ModelNet40 dataset, while Tables 7, 8 and 9 in the appendix show the performance when tested on different ShapeNet categories.

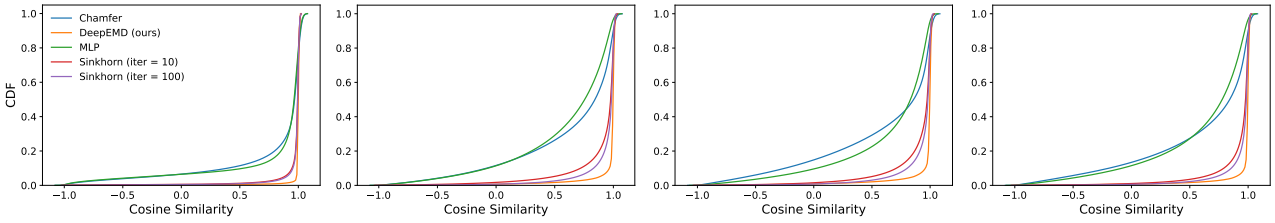
Table 1. Out-of-distribution (dataset) generalization for our models and comparison with other metrics (CD and Sinkhorn), tested on full validation split for ModelNet40 (with 40 categories). The models are trained on a single ShapeNet category. The reported numbers are averaged over these categories as well as four training seeds. The first six rows show distance estimation metrics (see § 3.2), while the last six rows correspond to matching estimation metrics. The arrows next to the metrics indicate whether higher (↑) values are better or lower (↓). Chamfer and Sinkhorn are deterministic, thus variances are not reported. Further, MLP does not provide accuracy and bipartiteness metrics.

MODEL	CHAMFER	SINKHORN	MLP	DEEPEMD
r (↑)	0.951	0.971	0.959 ± 0.011	$\mathbf{0.999} \pm 0.0$
ρ (↑)	0.935	0.988	0.945 ± 0.017	$\mathbf{0.999} \pm 0.0$
τ (↑)	0.792	**0.983**	0.819 ± 0.024	0.974 ± 0.002
$\text{RE}_{0.1}$ (↓)	0.03	0.057	0.009 ± 0.001	$\mathbf{0.005} \pm 0.002$
$\text{RE}_{0.5}$ (↓)	0.129	0.102	0.062 ± 0.005	$\mathbf{0.019} \pm 0.004$
$\text{RE}_{0.9}$ (↓)	0.321	0.2	0.257 ± 0.03	$\mathbf{0.04} \pm 0.004$
$\text{CS}_{0.1}$ (↑)	-0.067	0.824	-0.293 ± 0.047	$\mathbf{0.927} \pm 0.003$
$\text{CS}_{0.5}$ (↑)	0.834	0.986	0.684 ± 0.023	$\mathbf{1.0} \pm 0.0$
$\text{CS}_{0.9}$ (↑)	0.997	0.999	0.96 ± 0.003	$\mathbf{1.0} \pm 0.0$
ACCURACY (↑)	12.651	31.91	-	$\mathbf{56.38} \pm 0.604$
B (↑)	17.045	33.458	-	$\mathbf{70.401} \pm 0.672$
B_{corr} (↑)	6.544	19.615	-	$\mathbf{47.084} \pm 0.741$



(a) Chamfer (b) MLP (ours) (c) DeepEMD (ours) (d) Sinkhorn

Figure 3. Scatter plot for true vs. approximate EMD from different models/metrics on validation splits for Syn2D and ShapeNet datasets. DeepEMD (ours) consistently performs better across different categories as it has less dispersion. Sinkhorn algorithm becomes more accurate with more iterations. Also note that it encounters numerical errors for some examples.



(a) Syn2D (b) ShapeNet Airplane (c) ShapeNet Car (d) ShapeNet Chair

Figure 4. CDF of cosine similarity between true and estimated gradients for all points across all point clouds collected together on validation splits for Syn2D and ShapeNet datasets. The ideal cdf curve should have all the mass at cosine similarity 1. DeepEMD (ours) consistently outperforms all the other methods across different datasets.

Scaling number of points. Remarkably, size of point clouds during testing can differ greatly from those during training without degrading performance. Table 2 shows model performance for point cloud sizes ranging from 256 to 8196, while training was done with only 1024 points.

Computational Time and Complexity. Fig. 5 in the appendix compares the evaluation time for different models and metrics. DeepEMD achieves a significant speedup of about $100\times$ as compared to EMD and $40\times$ as compared to Sinkhorn with 100 iterations. This speedup becomes more pronounced on bigger point clouds as Hungarian algorithm takes $O(N^3)$ time vs. $O(N^2)$ for DeepEMD.

DeepEMD used as a loss. Training a SetVAE (Kim et al., 2021), as for any auto-encoder, requires a reconstruction loss to assess the quality of the learned representation. While the eventual goal would be to minimize the EMD, standard approach uses CD due to the prohibitive computation cost of calculating the EMD. Instead of CD we propose to use DeepEMD and demonstrate its utility as a reconstruction loss as compared to CD. The parameters of DeepEMD module are frozen during training of the SetVAE. Fig. 1 and Fig. 9 (appendix) shows the reconstruction on validation data achieved by SetVAE models trained with different reconstruction losses. DeepEMD consistently achieves lower reconstruction EMD as compared to CD. This is further verified from Fig. 6 (appendix) showing the distribution of true EMD between a point cloud and its reconstruction.

4. Conclusion and Future Work

We propose DeepEMD, a method for fast approximation of EMD, improving time complexity from $O(N^3)$ to $O(N^2)$. We demonstrated the effectiveness of DeepEMD in approximating the true EMD for various datasets. Further, we show that it estimates the gradients well, generalizes well for unseen point clouds (or distributions), and can be used for end-to-end training of point cloud autoencoders achieving faster convergence than CD surrogate. It would be interesting to explore fast transformer variants to further improve from the quadratic time complexity for future work. The extension to Wasserstein$-p$ metrics and other OT problems could also be interesting for various applications.

References

Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pp. 40–49. PMLR, 2018.

Blondel, M., Seguy, V., and Rolet, A. Smooth and sparse optimal transport. In *International conference on artificial intelligence and statistics*, pp. 880–889. PMLR, 2018.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Chuang, C.-Y., Jegelka, S., and Alvarez-Melis, D. Infoot: Information maximizing optimal transport. *arXiv preprint arXiv:2210.03164*, 2022.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Flamary, R., Courty, N., Tuia, D., and Rakotomamonjy, A. Optimal transport for domain adaptation. *IEEE Trans. Pattern Anal. Mach. Intell*, 1, 2016.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., et al. Pot: Python optimal transport. *The Journal of Machine Learning Research*, 22(1):3571–3578, 2021.

Genevay, A., Peyré, G., and Cuturi, M. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pp. 1608–1617. PMLR, 2018.

Huang, T., Yang, X., Zhang, J., Cui, J., Zou, H., Chen, J., Zhao, X., and Liu, Y. Learning to train a point cloud reconstruction network without matching. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I*, pp. 179–194. Springer, 2022.

Kim, J., Yoo, J., Lee, J., and Hong, S. Setvae: Learning hierarchical composition for generative modeling of set-structured data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15059–15068, 2021.

Luo, S. and Hu, W. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2837–2845, 2021.

Shu, D. W., Park, S. W., and Kwon, J. 3d point cloud generative adversarial network based on tree structured graph convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3859–3868, 2019.

Urbach, D., Ben-Shabat, Y., and Lindenbaum, M. Dpdist: Comparing point clouds using deep point cloud distance. In *European Conference on Computer Vision*, pp. 545–560. Springer, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wu, T., Pan, L., Zhang, J., Wang, T., Liu, Z., and Lin, D. Density-aware chamfer distance as a comprehensive metric for point cloud completion. *arXiv preprint arXiv:2111.12702*, 2021.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

Yang, G., Huang, X., Hao, Z., Liu, M.-Y., Belongie, S., and Hariharan, B. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4541–4550, 2019.

A. Results

A.1. Performance Measures

Distance Estimation. We visualize the true vs. predicted distance through scatter plots (Fig. 3), where we expect the data points to be close to $x = y$ line. We compare various correlation measures : linear correlation (r), Spearman correlation (ρ) and Kendall-Tau correlation (τ), to assess the quality of distance estimation. The Spearman and Kendall-Tau are rank-statistic based correlation measures, indicative of the correspondence between two rankings. Note that, correlation measures are useful metrics as they indicate appropriateness of the predicted metric as a distance measure, irrespective of their absolute values. Additionally we look at different quantiles (RE_n) of relative approximation error, which penalizes the difference between absolute values of the predicted and true distance.

Matching Estimation. In order to assess quality of the matching, we consider the cosine similarity between the true and predicted gradient. The true gradient of EMD is always along the matched point. We visualize the cumulative distribution function (cdf) of cosine similarities (Fig. 4), where we expect all the mass to be close to 1. We also look at different quantiles (CS_n) of the cosine similarity. We also consider accuracy which is computed as the average accuracy of matching source points to target points and vice-versa, bipartiteness (B) which is fraction of points with bipartite matching, and also bipartiteness-correctness (B_{corr}) which is fraction of points which are bipartite as well as matched correctly.

A.2. EMD Prediction

The MLP baseline seems to struggle a bit on ShapeNet Chair dataset. The higher dispersion in Chamfer (Fig. 3a) and Sinkhorn with 10 iterations (Fig. 3d, top) indicates poor EMD estimation. The approximation with Sinkhorn algorithm becomes more accurate with higher number of iterations (Fig. 3d, bottom), as expected. We summarize various metrics in Tables 4, 5 and 6 below.

A.3. Matching Estimation

Fig. 4 shows the cdf for cosine similarity between true and estimated gradients. The cdf has most mass at cosine similarity close to 1 with a very short tail and is never negative indicating that the estimated gradient is aligned with the true gradient for DeepEMD. This is particularly important when the model is used as a surrogate reconstruction loss. Ideally, the model should provide good estimate of the true gradient throughout training and more particularly in the very beginning when the reconstructions are very noisy, and also towards the end when reconstructions likely become very similar to the training distribution.

A.4. Out-of-distribution Generalization

The results in Tables 1, 7, 8 and 9 indicate that DeepEMD generalizes well when test and train data differ without any adaptation or fine-tuning. Further, the validation performance on a category of a model trained on another category (see Appendix for details) is very similar to the performance of the model trained on the same category. These quite remarkable behaviors point towards the network "meta-learning" in some way the matching algorithm. This is further strengthened by the results on scaling to different number of points during test time as shown in Table 2.

A.5. Scaling number of points

Table 2 shows performance of the model for test point cloud sizes ranging from 256 to 8196, while training was done with only 1024 points. Prediction of the metric itself (top 6 rows) does not degrade for all practical purposes. Regarding the matching estimation, directional measure of performance related to the cosine similarity (rows CS_n) do not degrade neither. We can notice degradation in accuracy based measures (last 3 rows) which is natural since the problem becomes difficult with increasing number of points N because of its combinatorial nature. For training when memory requirement is much higher due to backprop, we can use smaller number of points, and scale it up during inference without any fine-tuning.

A.6. Computational Time and Complexity

Fig. 5 compares the empirical evaluation time for different performance measures.

Table 2. Scaling number of points and out-of-distribution (scale) generalization for DeepEMD. The models are trained on a single ShapeNet category with 1024 points and tested on validation split of same category but with different number of points. Reported values are averaged over 4 training seeds. DeepEMD generalizes well to unseen number of points at test time without fine-tuning.

	⟵ Less # points than training ⟶			Trained	⟵ More # points than training ⟶		
# points	256	512	768	1024	2048	4096	8192
r	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.999 ± 0.0	0.999 ± 0.001
ρ	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.999 ± 0.0	0.998 ± 0.0
τ	0.985 ± 0.0	0.987 ± 0.0	0.988 ± 0.0	0.988 ± 0.001	0.986 ± 0.001	0.981 ± 0.002	0.974 ± 0.004
$\text{RE}_{0.1}$	0.002 ± 0.001	0.002 ± 0.001	0.004 ± 0.002	0.007 ± 0.003	0.012 ± 0.005	0.013 ± 0.007	0.014 ± 0.008
$\text{RE}_{0.5}$	0.01 ± 0.002	0.011 ± 0.002	0.014 ± 0.003	0.017 ± 0.005	0.027 ± 0.009	0.034 ± 0.013	0.04 ± 0.016
$\text{RE}_{0.9}$	0.026 ± 0.003	0.026 ± 0.003	0.029 ± 0.004	0.032 ± 0.005	0.042 ± 0.009	0.054 ± 0.013	0.066 ± 0.018
$\text{CS}_{0.1}$	0.94 ± 0.002	0.955 ± 0.002	0.961 ± 0.001	0.964 ± 0.001	0.967 ± 0.001	0.967 ± 0.001	-
$\text{CS}_{0.5}$	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	-
$\text{CS}_{0.9}$	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	-
Accuracy	72.348 ± 0.44	69.384 ± 0.383	66.901 ± 0.379	64.648 ± 0.404	57.588 ± 0.464	47.78 ± 0.51	35.274 ± 0.483
B	81.857 ± 0.755	80.101 ± 0.547	78.013 ± 0.507	75.896 ± 0.521	68.658 ± 0.584	58.109 ± 0.597	44.603 ± 0.734
B_{corr}	64.838 ± 0.756	61.558 ± 0.587	58.545 ± 0.547	55.719 ± 0.568	46.831 ± 0.618	35.053 ± 0.6	21.606 ± 0.469

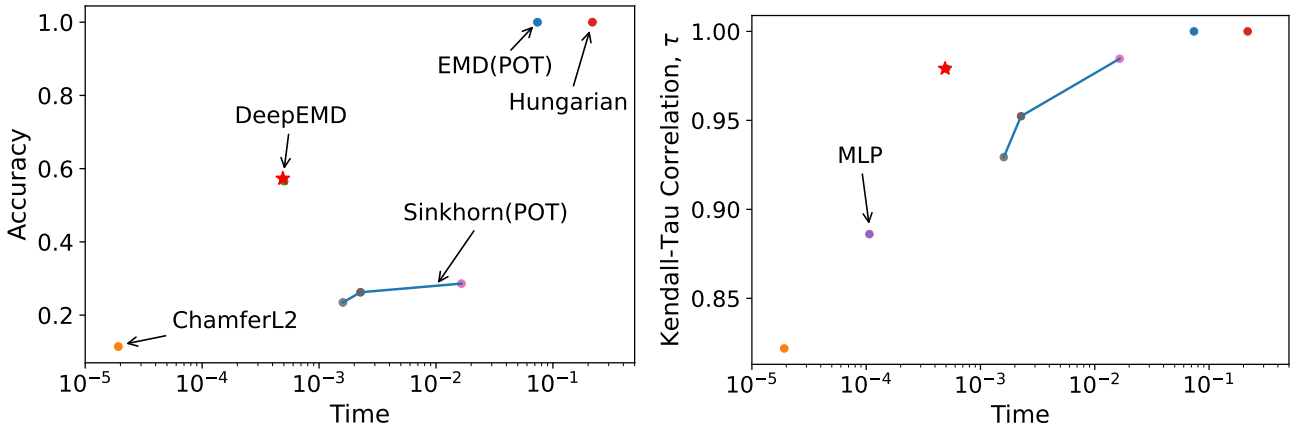


Figure 5. Comparison of empirical evaluation time and different performance measures : Accuracy (left) and Kendall-Tau correlation τ (right). We use Python Optimal Transport (POT) library for computing the Sinkhorn distances, and show metrics at different iterations (5, 10 and 100). DeepEMD is about $100\times$ and $40\times$ faster than Hungarian algorithm and Sinkhorn (100 iterations), respectively.

A.7. DeepEMD used as a loss

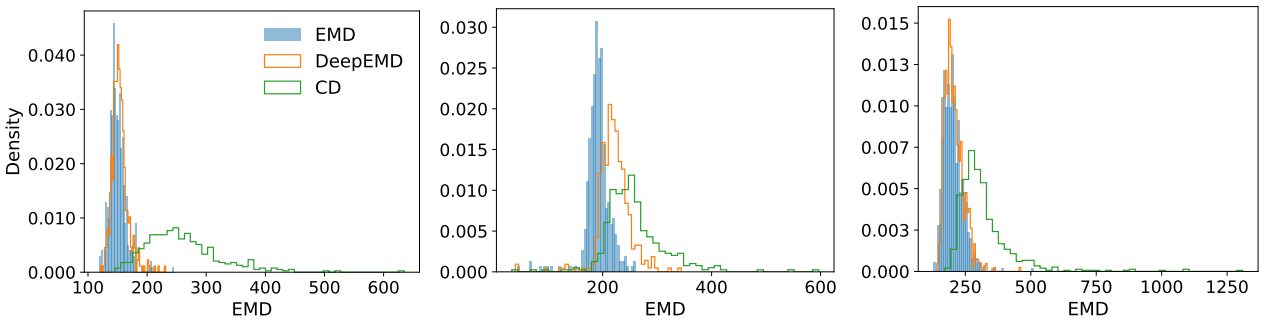
In our experiments, DeepEMD was trained separately on each category of ShapeNet dataset and the trained model was then used as a surrogate reconstruction loss for training a variational auto-encoder. We use SetVAE (Kim et al., 2021), a transformer based VAE adapted for point clouds and set-structured data. Fig. 6 below shows the distribution of true EMD between a point cloud and its reconstruction.

B. Optimal Transport and Wasserstein Distances

The Wasserstein-p metric between two probability distributions μ_X and ν_Y is defined as,

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \mathbf{E}_{(x,y)\sim\gamma} \|x - y\|_p \right)^{1/p}, \tag{2}$$

where $\Gamma(\mu, \nu)$ are all possible joint distributions where $X, Y \in \mathcal{D}$, $(\mathcal{D}, d)$ defines a metric space (here, $d = \|x - y\|_p$) and marginals satisfy $\int_{\mathcal{D}} \gamma(x, y)\mathrm{d}y = \mu(x)$ and $\int_{\mathcal{D}} \gamma(x, y)\mathrm{d}x = \nu(y)$. Note that, while the distance is useful in itself, the optimal transport plan $\gamma*$ is also interesting for some applications.



(a) ShapeNet Airplane (b) ShapeNet Car (c) ShapeNet Chair

Figure 6. Comparison of EMD between input and reconstructed point clouds from SetVAE trained with different reconstruction losses. The better model should have smaller reconstruction loss and thus mass close to zero in the histograms. DeepEMD (ours) is consistently better as compared to Chamfer loss and very similar to EMD loss.

Given samples from μ and ν, $W_p(\mu, \nu)$ can be computed by solving the optimal transport problem,

$$\gamma^* = \underset{\gamma \in \mathbb{R}_+^{m \times n}}{\operatorname{argmin}} \sum_{i,j} \gamma_{i,j} M_{i,j} \tag{3}$$

$$\text{s.t. } \gamma 1 = [\hat{\mu}]_m; \gamma^T 1 = [\hat{\nu}]_n; \gamma \geq 0 \tag{4}$$

where $[\hat{\mu}]_m$ and $[\hat{\nu}]_n$ represent binned histograms derived from samples from μ and ν with m and n bins respectively. M is a $m \times n$ distance or cost matrix, $M_{i,j}$ represents the cost $d(x, y)$ to transport mass from bin $[\hat{\mu}]_m^i$ to bin $[\hat{\nu}]_n^j$.

It is also possible to avoid binning and compute the Wasserstein distance directly from samples. The problem can then be formulated as,

$$\gamma^* = \underset{\gamma \in \mathbb{R}_+^{n \times n}}{\operatorname{argmin}} \sum_{i,j} \gamma_{i,j} M_{i,j} \tag{5}$$

$$\text{s.t. } \gamma 1 = 1; \gamma^T 1 = 1; \gamma \in \{0, 1\} \tag{6}$$

where $M_{i,j}$ now represents the cost of transporting point x_i to y_j. Each point is considered to be sampled i.i.d. from their respective distributions. Unlike the previous case, we can no longer transport a quanta of a point or mass i.e. fractional assignment is not meaningful, naturally leading to a bipartiteness constraint. This is exactly same as Eq (1) and the optimal transport plan for the problem is a linear sum assignment problem and can be computed using the hungarian algorithm.

C. Datasets

We train and evaluate on *Syn2D*, ShapeNet and ModelNet40 datasets.

Syn2D. We generate 2D synthetic point clouds by uniformly sampling 200 points on simple 2D shapes of circles and squares. The circles are generated by uniformly sampling the center and radius from $(0, 1]$. For the squares, we sample its center, rotation and scale uniformly from $[-0.5, 0.5]$, $[0, \frac{\pi}{2}]$, and $[0.5, 1]$, respectively. We refer to the synthetic dataset as *Syn2D* (Fig. 7).

ShapeNet. ShapeNet (Chang et al., 2015) is a richly-annotated, large-scale dataset of 3D shapes. We train and evaluate our models using point clouds from one of the three categories in the ShapeNet dataset : airplane, chair, and car. We sample 1024 points from the ShapeNet point clouds for training and evaluate on a range of point cloud sizes.

ModelNet40. (Chang et al., 2015) introduced the ModelNet project to provide a comprehensive and clean collection of $3D$ CAD models for objects, compiled using a list of the most common object categories. In our experiments, we evaluate our models trained with a single ShapeNet category on the ModelNet40 dataset. ModelNet40 has 40 different categories.

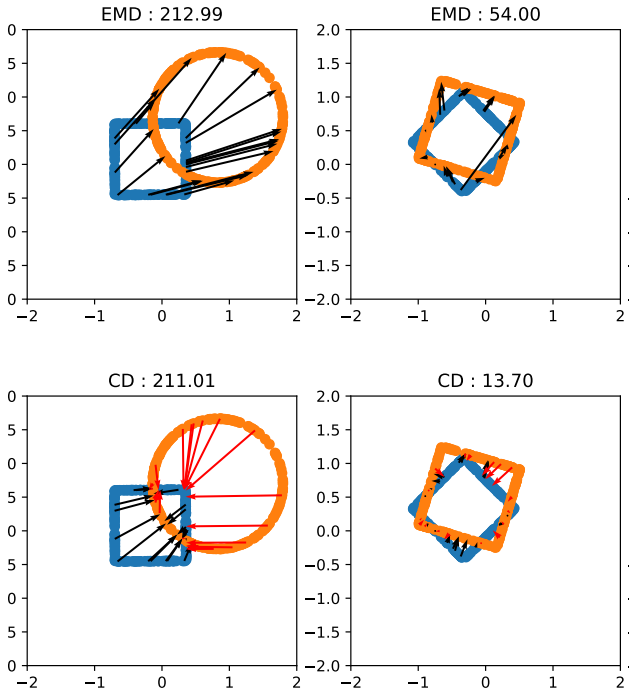


Figure 7. Example pairs of point clouds. The true earth mover's distance (EMD, top) and Chamfer distance (CD, bottom) are shown above each example. Arrows indicate the matching between the two point clouds under their respective metrics.

Table 3. Summary of datasets. The top five rows show statistics about the original point cloud datasets, while the bottom four rows show statistics about the pair point cloud dataset that was used for our evaluation and/or training.

	SYN2D	MODELNET40	SHAPENET	SHAPENET AIRPLANE	SHAPENET CAR	SHAPENET CHAIR
# CATEGORIES	2	40	55	1	1	1
FEATURE DIM	2	3	3	3	3	3
CARDINALITY	200	2048	15000	15000	15000	15000
# TRAIN SAMPLES	8000	9840	35708	2832	2458	4612
# VAL SAMPLES	2000	2468	5158	405	352	662
# TRAIN PAIR SAMPLES	20000	-	-	10000	10000	10000
TRAIN CARDINALITY	200	-	-	1024	1024	1024
# VAL PAIR SAMPLES	5000	2000	-	2000	2000	2000
VAL CARDINALITY	200	1024	-	256-8192	256-8192	256-8192

Table 3 provides a summary and statistics about of the datasets.

Further, we build pairs (U, V) of point clouds by randomly sampling pairs from the train or validation splits of point clouds datasets summarized in Table . We refer the first argument in the pair U as the *source* and the second argument V as *target*.

C.1. Augmentations

In order to improve generalization, we augment the datasets with point cloud pairs. We randomly sample a point cloud pair (U, V) from the dataset and another noisy point cloud N by randomly sampling points from $\mathcal{N}(0, 1)$ and scaling the whole point cloud by $\sigma \sim \mathcal{U}(0.1, 1.1)$. Further, we augment the point cloud pairs according to the following schemes:

- (U, V) : the originally sampled pair.

- (U, N) : target is replaced by the noisy point cloud.
- $(U, U + N)$: target is a corrupted version of U with additive noise N.
- $(U, \tilde{U} + N)$: $\tilde{U}$ denotes a point cloud which is similar to U. For *Syn2D*, we perturb the surface parameters (radius, scale, center, etc.) used for sampling U and sample points on the perturbed surface. For ShapeNet, we independently sample different set of points from the original surface.
- $(U, V + N)$: target is a corrupted version of a randomly sampled point cloud from the dataset.

The resulting dataset constitutes 20% samples from each of the above splits. Validation splits are generated randomly and independently for *Syn2D*, while for ShapeNet and ModelNet40, we use the validation split provided. We also augment the validation split with the same scheme as discussed above. The specifics of the pair point-cloud dataset is summarized in Table 3.

D. Models

Transformers. Let $X = [x_1, \cdots, x_m] \in \mathbb{R}^{m \times d_{\text{model}}}$ be the input sequence (or set) of m vectors. A transformer layer performs the following computation (Vaswani et al., 2017) :

$$X' = \text{LN}\left(X + \text{Multihead}(X)\right)$$
$$t_l(X) = \text{LN}\left(X' + \text{FFN}(X')\right)$$

where LN and FFN stand for layer norm and feed-forward network, respectively. Multihead denotes a multi-head attention layer which allows the model to jointly attend to information from different representation subspaces at different positions. It consists of a stack of H scaled dot-product attention layers and computes key, query and value matrices, followed by a softmax as follows :

$$K_h = XW_h^K, \quad Q_h = XW_h^Q, \quad V_h = XW_h^V$$
$$A_h = \text{soft-max}\left(\frac{Q_h {K_h}^\top}{d_k}\right) V_h$$
$$\text{MultiHead}(X) = \bar{A} = \text{concat}\left(A_1, \cdots, A_H\right) W_O$$

The final output of the encoder can be written as a composition:

$$t(X) = t_N\left(t_{N-1}\left(\ldots\left(t_1(X)\right)\right)\right)$$

MLP. We propose a simple MLP baseline composed of a point-wise MLP backbone, followed by a prediction head which is also a MLP (see figure 8). The backbone MLP takes a point cloud and returns an embedding $e \in \mathbb{R}^d$ as,

$$e_u = \sum_{i=1}^{n} g(u_i), \quad e_v = \sum_{j=1}^{n} g(v_i) \tag{7}$$

The prediction head then produces the final prediction as,

$$\hat{d} = h\left(e_u \oplus e_v\right) + h\left(e_v \oplus e_u\right), \tag{8}$$

where $\oplus$ denotes vector concatenation. Both g and h are composed of sequential linear layers with ReLU non-linearity between layers. The embeddings are permutation equivariant because of the sum aggregation which does not depend on the ordering of points. Further, we concatenate the embeddings both ways as in Eq. (8), which makes the mapping symmetric. We train the model with mean-squared error loss, $l = (d - \hat{d})^2$. Since the model does not predict the matching, we can interpret it from the direction of the gradient of a point $\delta v_j = \left[\frac{\partial \hat{d}}{\partial V}\right]_j$, e.g., by taking cosine similarity between δv_j and $u_i - v_j$, where u_i is the point matching to v_j from EMD.

The baseline MLP model predicts directly the distance and it does not use the matching information. It has about $110K$ parameters in total. The point-wise MLP backbone $g(.)$ is composed of three hidden layer of sizes $4, 8$ and 16, with ReLU non-linearity. It outputs a single embedding of dimension 128 for each point cloud after aggregating the point level features. The embedding of the point clouds are then concatenated and passed to the prediction head which is also an MLP with four hidden layers of sizes $256, 128, 64$ and 16, and outputs a single scalar which is interpreted as the predicted distance.

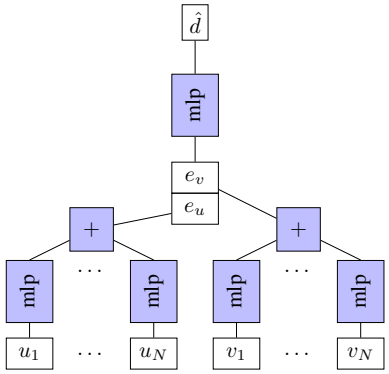


Figure 8. The MLP model predicts directly an estimate $\hat{d}$ of the EMD.

DeepEMD. We use a transformer encoder backbone which transforms the raw input point clouds into contextualized point level features. The transformer encoder is followed by the output layer which computes the queries and keys and finally the attention matrix which is interpreted as the matching as explained in § 2. The model constitutes about $803K$ learnable paramters, with 8 transformer encoder layers, each with 6 heads. The latent dimensions ($d_{\text{model}}, d_{\text{keys}}$, etc.) for each layer were all set to 78.

We use the ADAM optimizer with a constant learning rate of 0.001 for DeepEMD and 0.0001 for the MLP.

E. Sinkhorn Distance

The Sinkhorn distance (Cuturi, 2013) considers a regularized OT optimization problem :

$$\gamma^* = \underset{\gamma \in \mathbb{R}_+^{m \times n}}{\operatorname{argmin}} \sum_{i,j} \gamma_{i,j} M_{i,j} + \lambda \Omega(\gamma)$$
$$\text{s.t. } \gamma 1 = [\hat{\mu}]_m; \gamma^T 1 = [\hat{\nu}]_n; \gamma \geq 0$$

where λ is the regularization coefficient and $O(\gamma) = \sum_{i,j} \gamma_{i,j} \log(\gamma_{i,j})$ is a entropy regularization term which makes the optimization problem smooth and strictly convex allowing for optimization procedures such as the Sinkhorn-Knopp algorithm. In this paper, we use the Python Optimal Transport (POT) python library (Flamary et al., 2021) for computing the Sinkhorn distances. It is an iterative algorithm and can be evaluated in $O(N^2)$ time complexity.

F. Extended Results

F.1. Distance and Matching Estimation

We compare performance of different models and metrics in Table 4. The models were trained on a single ShapeNet category and evaluated on the validation split of the same category. The measures are averaged over all training categories as well as four training seeds.

Tables 5 and 6 show the per-category performance comparison.

F.2. Out-of-distribution generalization

Table 7 shows the out-of-distribution generalization performance for our models. The trained model on a particular ShapeNet category is evaluated on the validation split of other ShapeNet categories. The numbers are averaged over these other

Table 4. Performance comparison of different metrics and models. The models are trained on a single ShapeNet category and evaluated on the test split of the same category. The reported numbers are averaged over all categories and four training seeds. The first six rows show distance estimation metrics (see § 3.2), while the last six rows correspond to matching estimation metrics. The arrows next to the metrics indicate whether higher (↑) or lower (↓) values are better. Chamfer and Sinkhorn are deterministic, thus variances are not reported. Further, our MLP model does not provide accuracy and bipartiteness metrics.

MODEL	CHAMFER	SINKHORN	MLP (OURS)	DEEPEMD (OURS)
r (↑)	0.963	0.995	0.998 ± 0.0	$\mathbf{1.0} \pm 0.0$
ρ (↑)	0.953	0.997	0.998 ± 0.001	$\mathbf{1.0} \pm 0.0$
τ (↑)	0.827	0.987	0.966 ± 0.003	$\mathbf{0.988} \pm 0.001$
$\text{RE}_{0.1}$ (↓)	0.023	0.051	$\mathbf{0.002} \pm 0.0$	0.007 ± 0.003
$\text{RE}_{0.5}$ (↓)	0.109	0.106	$\mathbf{0.015} \pm 0.001$	0.017 ± 0.005
$\text{RE}_{0.9}$ (↓)	0.31	0.271	0.076 ± 0.006	$\mathbf{0.032} \pm 0.005$
$\text{CS}_{0.1}$ (↑)	-0.173	0.831	-0.034 ± 0.049	$\mathbf{0.964} \pm 0.001$
$\text{CS}_{0.5}$ (↑)	0.85	0.986	0.798 ± 0.018	$\mathbf{1.0} \pm 0.0$
$\text{CS}_{0.9}$ (↑)	0.998	0.999	0.974 ± 0.003	$\mathbf{1.0} \pm 0.0$
ACCURACY (↑)	11.677	28.407	-	$\mathbf{64.648} \pm 0.404$
B (↑)	17.784	31.889	-	$\mathbf{75.896} \pm 0.521$
B_{corr} (↑)	5.626	16.658	-	$\mathbf{55.719} \pm 0.568$

Table 5. Per-category distance estimation performance measures of different models and metrics when train and test category are same. The reported number are averaged over four training seeds.

		r	ρ	τ	$\text{RE}_{0.1}$	$\text{RE}_{0.5}$	$\text{RE}_{0.9}$
TRAIN_CATE	MODEL/METRIC						
AIRPLANE	CHAMFER	0.9797	0.9647	0.8519	0.0141	0.0962	0.3018
	DEEPEMD	$\mathbf{0.9998} \pm 0.0$	$\mathbf{0.9997} \pm 0.0$	0.9879 ± 0.001	0.0039 ± 0.0014	0.0142 ± 0.0031	$\mathbf{0.0306} \pm 0.004$
	MLP	0.9992 ± 0.0001	0.9986 ± 0.0001	0.9722 ± 0.0018	$\mathbf{0.002} \pm 0.0003$	$\mathbf{0.0125} \pm 0.0018$	0.0772 ± 0.0038
	SINKHORN	0.9998	0.9997	**0.9881**	0.0496	0.1104	0.2984
CAR	CHAMFER	0.9675	0.9564	0.8318	0.035	0.1167	0.324
	DEEPEMD	$\mathbf{0.9997} \pm 0.0001$	$\mathbf{0.9998} \pm 0.0$	$\mathbf{0.9891} \pm 0.0006$	0.006 ± 0.0047	0.0156 ± 0.0068	$\mathbf{0.0302} \pm 0.0097$
	MLP	0.9993 ± 0.0002	0.9988 ± 0.0004	0.9746 ± 0.0035	$\mathbf{0.0018} \pm 0.0002$	$\mathbf{0.0112} \pm 0.001$	0.0625 ± 0.012
	SINKHORN	0.991	0.9941	0.9854	0.0531	0.1124	0.2986
CHAIR	CHAMFER	0.9431	0.9382	0.7976	0.0192	0.1133	0.3037
	DEEPEMD	$\mathbf{0.9997} \pm 0.0$	$\mathbf{0.9997} \pm 0.0001$	0.9866 ± 0.0013	0.0103 ± 0.0088	0.0225 ± 0.0118	$\mathbf{0.0356} \pm 0.012$
	MLP	0.9965 ± 0.0013	0.9959 ± 0.0015	0.9504 ± 0.0079	$\mathbf{0.0037} \pm 0.0006$	$\mathbf{0.0214} \pm 0.0023$	0.0881 ± 0.0124
	SINKHORN	0.9942	0.9969	**0.9886**	0.049	0.0955	0.2167

categories as well as four training seeds. Tables 8 and 9 show the performance in the same setting but for each test category separately.

F.3. DeepEMD as a loss

Fig. 9 shows more samples with the input point cloud and the reconstructed output from SetVAE when trained with EMD, Chamfer or DeepEMD as the reconstruction loss.

Table 6. Per-category matching estimation performance measures of different models and metrics when train and test category are same. The reported number are averaged over four training seeds.

		$CS_{0.1}$	$CS_{0.5}$	$CS_{0.9}$	ACCURACY	B	B_CORR
TRAIN_CATE	MODEL/METRIC						
AIRPLANE	CHAMFER	-0.0813	0.8446	0.9973	10.1768	16.7437	4.7461
	DEEPEMD	$\mathbf{0.9643} \pm 0.0027$	$\mathbf{1.0} \pm \mathbf{0.0}$	$\mathbf{1.0} \pm 0.0$	$\mathbf{61.9119} \pm 0.9043$	$\mathbf{73.3128} \pm 1.1746$	$\mathbf{52.2082} \pm 1.246$
	MLP	-0.0492 ± 0.0533	0.7766 ± 0.0263	0.9722 ± 0.003	-	-	-
	SINKHORN	0.8314	0.9871	0.9994	25.2956	29.018	13.9732
CAR	CHAMFER	-0.246	0.8615	1.0	13.7079	20.7186	6.8617
	DEEPEMD	$\mathbf{0.9585} \pm 0.0025$	$\mathbf{1.0} \pm \mathbf{0.0}$	$\mathbf{1.0} \pm 0.0$	$\mathbf{67.7243} \pm 0.6554$	$\mathbf{78.7286} \pm 0.797$	$\mathbf{59.6723} \pm 0.9245$
	MLP	0.017 ± 0.0652	0.8388 ± 0.0187	0.9804 ± 0.0029	-	-	-
	SINKHORN	0.8043	0.9845	0.9993	31.1621	35.5971	19.3105
CHAIR	CHAMFER	-0.1929	0.8442	0.9976	11.145	15.8883	5.2694
	DEEPEMD	$\mathbf{0.9703} \pm 0.0007$	$\mathbf{1.0} \pm \mathbf{0.0}$	$\mathbf{1.0} \pm 0.0$	$\mathbf{64.3079} \pm 0.4712$	$\mathbf{75.6459} \pm 0.657$	$\mathbf{55.2757} \pm 0.7011$
	MLP	-0.0695 ± 0.1211	0.7793 ± 0.0438	0.97 ± 0.0072	-	-	-
	SINKHORN	0.8558	0.9876	0.9992	28.7644	31.0521	16.69

Table 7. Out-of-distribution (category) generalization for our models and comparison with other metrics (Chamfer and Sinkhorn). The models are trained on a single ShapeNet category and evaluated on other ShapeNet categories. The reported numbers are averaged over these categories as well as four training seeds. The first five rows show distance estimation metrics (see § 3.2), while the last five rows correspond to matching estimation metrics. The arrows next to the metrics indicate whether higher (↑) or lower (↓) values are better.

MODEL	MLP (OOD)	MLP	DEEPEMD (OOD)	DEEPEMD
r (↑)	0.98 ± 0.019	0.998 ± 0.001	0.999 ± 0.001	1.0 ± 0.0
ρ (↑)	0.976 ± 0.02	0.998 ± 0.001	0.999 ± 0.0	1.0 ± 0.0
τ (↑)	0.886 ± 0.04	0.966 ± 0.004	0.977 ± 0.003	0.988 ± 0.001
$RE_{0.1}$ (↓)	0.014 ± 0.003	0.002 ± 0.0	0.009 ± 0.009	0.007 ± 0.005
$RE_{0.5}$ (↓)	0.065 ± 0.018	0.015 ± 0.002	0.024 ± 0.012	0.017 ± 0.007
$RE_{0.9}$ (↓)	0.319 ± 0.132	0.076 ± 0.009	0.05 ± 0.011	0.032 ± 0.008
$CS_{0.1}$ (↑)	-0.208 ± 0.089	-0.034 ± 0.074	0.933 ± 0.004	0.964 ± 0.002
$CS_{0.5}$ (↑)	0.714 ± 0.047	0.798 ± 0.027	1.0 ± 0.0	1.0 ± 0.0
$CS_{0.9}$ (↑)	0.963 ± 0.006	0.974 ± 0.004	1.0 ± 0.0	1.0 ± 0.0
ACCURACY (↑)	-	-	54.35 ± 1.16	64.648 ± 0.606
B (↑)	-	-	67.922 ± 1.343	75.896 ± 0.782
B_{corr} (↑)	-	-	44.293 ± 1.445	55.719 ± 0.851

Table 8. Per-category distance estimation performance measures of different models and metrics when train and test category are different. The reported number are averaged over four training seeds.

		r	ρ	τ	$RE_{0.1}$	$RE_{0.5}$	$RE_{0.9}$
TRAIN_CATE	TEST_CATE						
AIRPLANE	AIRPLANE	0.9998 ± 0.0	0.9997 ± 0.0	0.9879 ± 0.001	0.0039 ± 0.0014	0.0142 ± 0.0031	0.0306 ± 0.004
	CAR	0.9989 ± 0.0006	0.9997 ± 0.0001	0.9864 ± 0.001	0.0035 ± 0.0005	0.016 ± 0.0036	0.0308 ± 0.0039
	CHAIR	0.9981 ± 0.0003	0.9983 ± 0.0003	0.9689 ± 0.0016	0.0044 ± 0.0011	0.0212 ± 0.0033	0.0491 ± 0.0048
CAR	AIRPLANE	0.9993 ± 0.0002	0.9989 ± 0.0003	0.9766 ± 0.0017	0.0092 ± 0.0065	0.0256 ± 0.011	0.0634 ± 0.0114
	CAR	0.9997 ± 0.0001	0.9998 ± 0.0	0.9891 ± 0.0006	0.006 ± 0.0047	0.0156 ± 0.0068	0.0302 ± 0.0097
	CHAIR	0.9978 ± 0.0001	0.9981 ± 0.0002	0.9671 ± 0.0017	0.0045 ± 0.0019	0.0206 ± 0.006	0.0525 ± 0.0022
CHAIR	AIRPLANE	0.999 ± 0.0004	0.9983 ± 0.0008	0.9751 ± 0.0047	0.0164 ± 0.0114	0.034 ± 0.0127	0.0631 ± 0.0092
	CAR	0.9987 ± 0.0007	0.9996 ± 0.0001	0.9867 ± 0.0007	0.0141 ± 0.0116	0.0275 ± 0.014	0.0422 ± 0.0139
	CHAIR	0.9997 ± 0.0	0.9997 ± 0.0001	0.9866 ± 0.0013	0.0103 ± 0.0088	0.0225 ± 0.0118	0.0356 ± 0.012

Table 9. Per-category matching estimation performance measures of different models and metrics when train and test category are different. The reported number are averaged over four training seeds.

		$CS_{0.1}$	$CS_{0.5}$	$CS_{0.9}$	ACCURACY	B	B_CORR
TRAIN_CATE	TEST_CATE						
AIRPLANE	AIRPLANE	0.9643 ± 0.0027	1.0 ± 0.0	1.0 ± 0.0	61.9119 ± 0.9043	73.3128 ± 1.1746	52.2082 ± 1.246
	CAR	0.9347 ± 0.004	1.0 ± 0.0	1.0 ± 0.0	61.4086 ± 0.9383	72.6142 ± 1.3357	51.6545 ± 1.3383
	CHAIR	0.926 ± 0.0045	0.9994 ± 0.0001	1.0 ± 0.0	48.9237 ± 0.9275	63.1441 ± 1.0583	38.1275 ± 1.1302
CAR	AIRPLANE	0.9279 ± 0.0027	0.9997 ± 0.0	1.0 ± 0.0	50.7472 ± 1.1176	66.2 ± 1.0652	40.6049 ± 1.3128
	CAR	0.9585 ± 0.0025	1.0 ± 0.0	1.0 ± 0.0	67.7243 ± 0.6554	78.7286 ± 0.797	59.6723 ± 0.9245
	CHAIR	0.9218 ± 0.0035	0.9994 ± 0.0001	1.0 ± 0.0	47.7712 ± 1.1373	63.8324 ± 1.1691	37.6786 ± 1.2958
CHAIR	AIRPLANE	0.9401 ± 0.0031	0.9998 ± 0.0	1.0 ± 0.0	53.7969 ± 0.8777	67.1421 ± 1.0913	43.3759 ± 1.0926
	CAR	0.945 ± 0.0006	1.0 ± 0.0	1.0 ± 0.0	63.4508 ± 0.5761	74.5965 ± 0.7858	54.3188 ± 0.8313
	CHAIR	0.9703 ± 0.0007	1.0 ± 0.0	1.0 ± 0.0	64.3079 ± 0.4712	75.6459 ± 0.657	55.2757 ± 0.7011

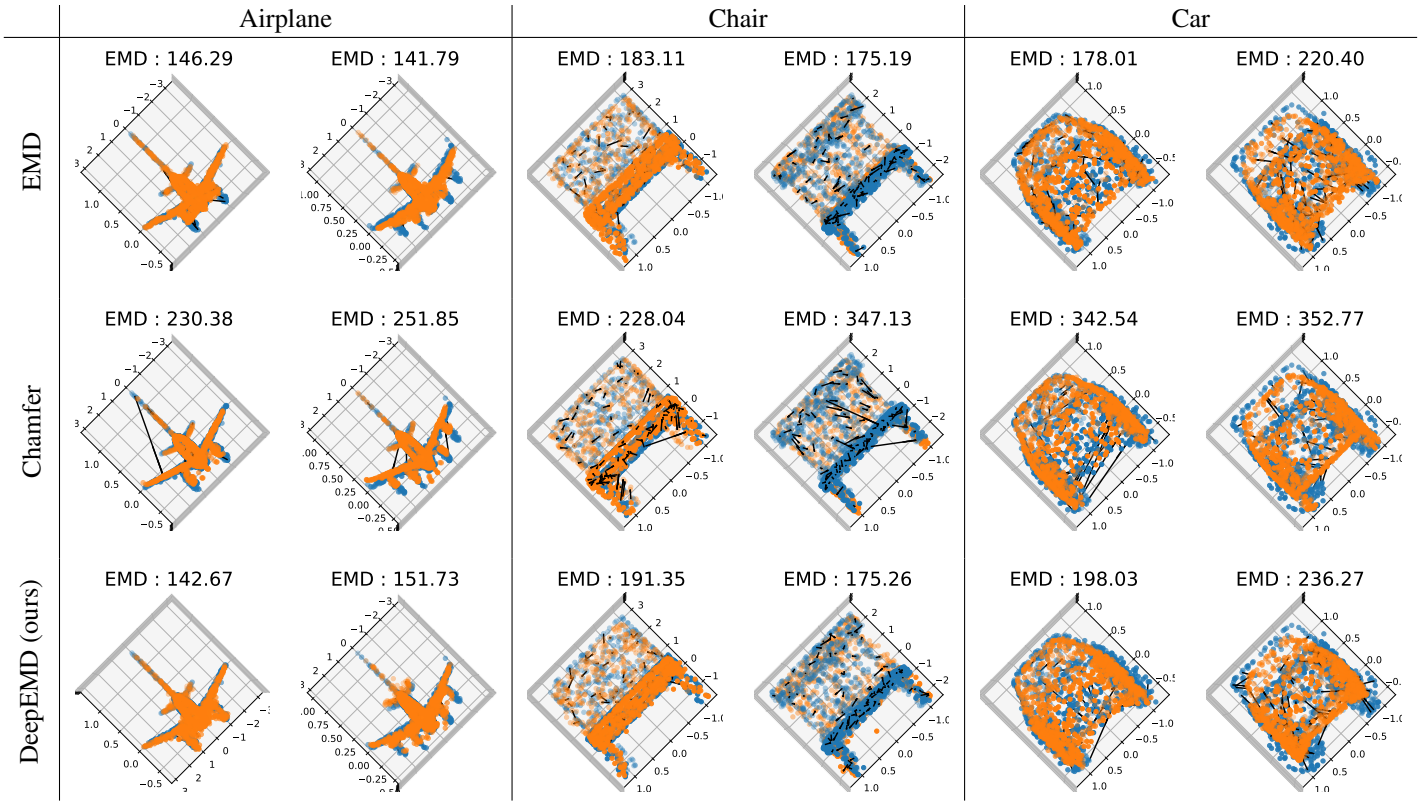


Figure 9. Reconstruction on validation data with SetVAE trained with differnt reconstruction losses : EMD (top), Chamfer (middle) and DeepEMD surrogate (bottom). Training with DeepEMD as a loss consistently achieves lower reconstruction EMD as compared to Chamfer loss.