SUPPLEMENTARY MATERIAL FOR "GEOMETRY REGULARIZED POINT CLOUD AUTOENCODER"

1. INTRODUCTION

In this supplementary material, we present more discussions on our experimental setup, as well as provide more experimental results and analysis on GRAE. We additionally include one more downstream task—object part segmentation—to demonstrate the usefulness of our proposal.

2. MORE EXPERIMENTAL SETTINGS AND DISCUSSIONS

2.1. EMD Computation

We hereby discuss more on the existing implementations of EMD. For applications in 2D image processing literature, an excellent, if dated, discussion is provided in [13], Section 2.1. In summary, the optimization problem underlying EMD can be framed as an assignment problem which is solvable using the Hungarian algorithm [7] in $O(N^3)$ time, while $O(\varepsilon)$ approximations (either using Sinkhorn, i.e., entropic approximation [2], or Bertsekas' auction algorithm [3]) can be computed in $O(N^2/\varepsilon)$ time. Recent theoretical refinements for approximation algorithms have made incremental improvements in the above rates (cf. [8]).

The two most common implementations of EMD in point cloud literature are those provided by [4] and [9] respectively. The first is faster in practice (although not comparable to CD). However, it is considerably divergent from Bertsekas' auction algorithm, which it claims to follow, and has no known theoretical guarantees. The latter is known to be an iterative algorithm that produces approximate solutions but does not have convergence guarantees. Newer articles using these implementations (e.g., [15]) in some cases report adverse findings for EMD in comparison to CD as a training loss due to the inability to find a near-optimal matching for moderately-sized point clouds.

Our EMD computation uses the entropic approximation-based multiscale Sinkhorn algorithm provided by the `geomloss` library [5], which well approximates the true EMD (with $O(\varepsilon)$ error) while having reasonable computational cost of $O(N^2/\varepsilon)$, and is also a GPU implementation.

While the entropic approximation algorithm used in `geomloss` is slower in practice compared to the implementation in [9], it produces high-quality reconstructed point clouds with no artifacts as observed with the latter [4]. We use the penalty parameters $\varepsilon = 0.1$ for training and $\varepsilon = 0.01$ for testing.

In our experiments, EMD as a training loss has an advantage, achieving comparable CD loss to CD-trained networks, and much better EMD loss. This supports the view in the prevailing literature that EMD, owing to its richer geometric properties, is a better loss for point cloud tasks compared to CD [1, 9], though the computational cost is a great concern.

2.2. Computational Complexity

The primary computational burden of our approach comes from the calculation of covariance matrices over nearest-neighbor graphs. However, as is standard in other point cloud methods utilizing local covariance matrices (e.g., ([16])), we select a set of nearest neighbor graph sizes and pre-compute local covariance matrices for the given sizes. For inverse computations, we further pre-process the covariance matrices and store only their Cholesky factors. When training with randomly rotated point clouds, since the rotation matrix is known at train time, the Cholesky factors can be adjusted by matrix multiplication, which is a much cheaper operation to do online compared to the Cholesky factorization step itself.

With these pre-processing steps, our approach (GRAE) emulates the performance of EMD in reconstructing the correct point distribution, with a much lower computational cost. Particularly, CD-based networks take roughly 3.5 hrs to train on average, while GRAE and GSW take about 1.5 times as much, compared to EMD which takes 10 times as much and is infeasible for larger point cloud sizes than those considered here. Additionally, it visually reproduces fine local features better than either EMD or GSW, which is also reflected in lower CD metrics. We note that all the experiments are performed on a workstation with an NVIDIA Quadro RTX 5000 GPU (16 GB) and six Intel Silver 4214 CPUs (2.2 GHz). Aside from the training complexity stated earlier, each of the reconstruction experiments can be finished within one day, while each of the segmentation and classification experiments can be finished within 12 hrs.

Table 1: Visualization of reconstructed point clouds under various geometry learning approaches and network architectures. Note that *random rotation* is applied to both training and testing. Our GRAE leads to higher geometric fidelity compared to other approaches.

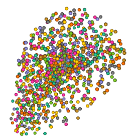
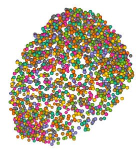
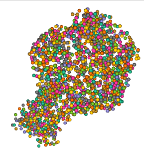
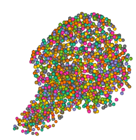
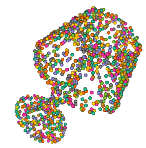
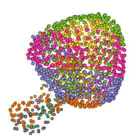
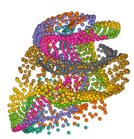
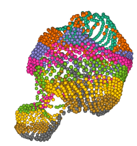
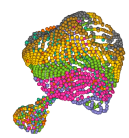
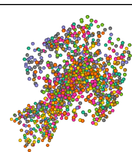
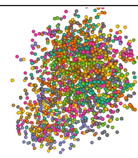
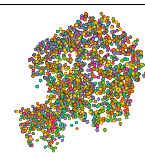
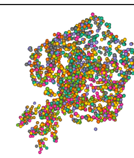
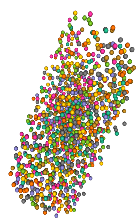
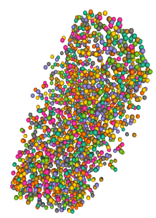
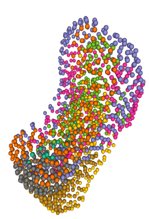
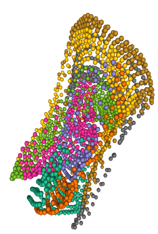
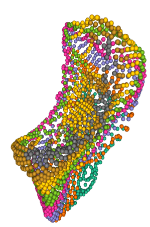
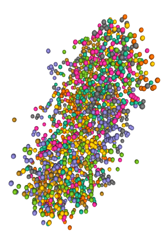
Data.	Architecture	CD	GSW	EMD	GRAE (Ours)	Ground-truth
SN	PointMLP+L.GAN					
	PointNet+Folding					
	PointCapsNet					
MN	PointMLP+L.GAN					
	PointNet+Folding					
	PointcapsNet					

Table 2: Classwise segmentation accuracies for various training losses.

Class	Airplane					Car				
Tr. Loss	PN+Fold	PN+MLP	PM+MLP	PCN	Avg.	PN+Fold	PN+MLP	PointMLP	PCN	Avg.
CD	78.99%	77.52%	84.16%	**77.90%**	79.64%	75.83%	74.08%	81.22%	**80.82%**	77.99%
EMD	79.15%	78.69%	**84.85%**	**77.42%**	80.03%	75.44%	75.49%	**83.52%**	80.23%	78.67%
GSW	**80.14%**	**81.30%**	75.56%	71.70%	77.17%	**78.31%**	**81.29%**	70.08%	78.66%	77.09%
GRAE	**80.69%**	**79.67%**	**85.68%**	75.32%	**80.34%**	**78.25%**	**76.20%**	**83.48%**	**80.48%**	**79.60%**
Class	Chair					Table				
CD	87.96%	87.81%	91.17%	**89.92%**	89.21%	**89.67%**	88.26%	92.81%	**91.32%**	90.52%
EMD	88.16%	88.27%	**91.35%**	89.40%	89.29%	89.59%	89.59%	**92.86%**	**90.24%**	90.57%
GSW	**88.68%**	**88.93%**	89.58%	86.73%	88.48%	88.68%	**89.89%**	91.24%	88.01%	89.46%
GRAE	**89.30%**	**88.85%**	**91.95%**	**89.59%**	**89.92%**	**90.01%**	**90.22%**	**92.86%**	89.77%	**90.72%**

Table 3: Overall segmentation accuracy

Network / Tr. Loss	PN+Fold	PN+MLP	PM+MLP	PCN	Avg
CD	74.87%	71.62%	**82.37%**	**78.16%**	76.76%
EMD	74.17%	72.25%	82.36%	**77.82%**	76.65%
GRAE	**77.16%**	**72.35%**	**84.04%**	75.58%	**77.28%**
GSW	**75.43%**	**79.76%**	78.57%	71.89%	76.41%

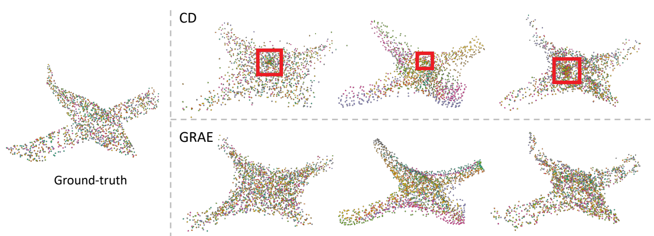


Fig. 1: Point collapse of training with CD is resolved by the proposed GRAE. The three columns on the right panel are reconstructions from PointMLP+LatentGAN, PointNet+Folding, and PointCapsNet, respectively.

3. MORE EXPERIMENTAL RESULTS

3.1. More Visualizations

We present more reconstruction renderings to further validate the effectiveness of our proposed GRAE. Similar to the experiments in the main text, we compare GRAE with CD, EMD [4], and GSW [6], and focus on the following autoencoder architectures: PointMLP(PM)+MLP [11, 1], PointNet(PN)+Folding [16, 14], and PointCapsNet (PCN) [18].

The renderings of the reconstructed point clouds from both the ShapeNet and the ModelNet datasets are visualized in Table 1. We clearly see that compared to other methods, our proposed GRAE retains more geometry details, presenting reconstructions with higher fidelity. Again, we emphasize that our experiments apply *random rotation* to both training and testing which makes the geometry reconstruction task much more challenging.

3.2. On Point Collapse

As discussed in [1, 12], using CD for training causes an issue called point collapse—a disproportionate number of points (compared to the input) are clustered at a certain median location in the reconstruction. In the first row of Fig. 1, an example of point collapse is presented where the regions undergoing the point collapse issue are also highlighted. Interestingly, our proposed GRAE addresses the issue and leads to a more natural point distribution, as seen in the second row of Fig. 1. Please refer to the supplementary material for more ablation studies and comparisons.

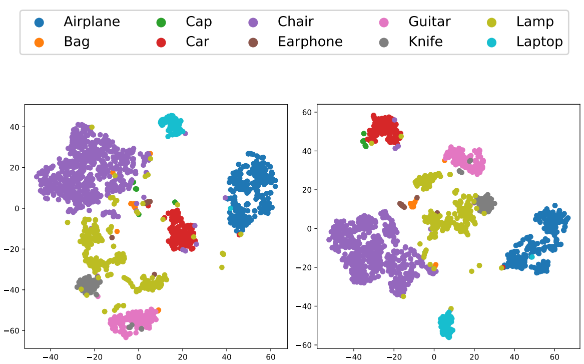
3.3. Segmentation

To demonstrate the validity of the codewords produced by our proposal, we also carry out a segmentation task based on intermediate point encodings produced by autoencoders trained using our proposal and other candidate losses. In this task, DGCNN-based networks are not used owing to the fact that they do not directly produce point encodings. For PointNet, PointCapsuleNet, and FoldingNet, the output of the point-wise MLP layers is taken as the point encoding, while for PointMLP, the point embedding layer of the encoder is used [10]. We emphasize here that the goal of the experiment is not to demonstrate the superiority of a segmentation method based on point embeddings. Rather, we wish to determine the validity of point embeddings generated by point autoencoders as bonafide representatives of the information contained in the points, and as such, we use point embeddings generated by the autoencoder version of PointCapsuleNet, and not the dedicated segmentation network [17] also propose.

In Table 2, we present classification accuracies for downstream segmentation networks trained class-wise on the 4 largest classes in the ShapeNetPart dataset. The same training and testing setup as classification is followed, except that we restrict the training and testing dataset to a single class for the segmentation network. GRAE ranks among the top 2 training losses in segmentation accuracy for all pairs of network architecture and class except one, and has the best average accuracy in all classes. PointMLP again has the best results, and is improved substantially by GRAE.

Table 4: Performance of different variants of GRAE. GRAE_F—GRAE with fixed neighborhood size $k = 8$. GRAE_H—GRAE without the term LMD_rec. Metrics are reported on a 10^{-2} scale.

Metric	Dataset	ShapeNet			ModelNet		
	Architecture	GRAE_F	GRAE_H	GRAE	GRAE_F	GRAE_H	GRAE
CD	PN+MLP	5.227	4.251	**3.848**	6.628	5.361	**4.553**
	DGCNN+MLP	5.861	5.430	**4.904**	**5.861**	6.016	5.864
	PM+MLP	5.429	4.439	**4.005**	6.669	5.354	**4.456**
	PN+Fold	3.825	3.813	**3.482**	4.614	4.477	**4.073**
	PCN	4.793	3.937	**3.754**	5.862	4.694	**4.185**
EMD	PN+MLP	0.840	0.894	**0.730**	0.739	0.704	**0.575**
	DGCNN+MLP	1.335	1.584	**1.203**	1.098	1.262	**1.025**
	PM+MLP	0.906	0.992	**0.869**	0.724	0.714	**0.631**
	PN+Fold	0.790	0.638	**0.524**	0.888	0.604	**0.508**
	PCN	1.338	1.201	**1.115**	1.209	1.049	**0.954**



(a) CD (b) GRAE

Fig. 2: Visualize the codewords with t-SNE in the 2D space. The codewords are generated with PointNet [14] that is trained under the PointNet+LatentGAN design.

Table 3 presents the results of running the segmentation experiment with all classes combined. Here, the segmentation accuracy numbers are smaller since the problem is now a 50-class classification with point classes from all types of objects pooled together, and the variability in accuracy numbers is also higher among different architectures. However, GRAE still has the best average accuracy. The top two numbers for each architecture are in bold.

3.4. Visualization of codewords

To gain a deeper understanding, we also visualize the codewords from the first 10 classes of ShapeNet with t-SNE in 2D, as is shown in Fig. 2 The codewords are generated with the PointNet encoder [14] trained with the PointNet+MLP architecture. We see that the codewords generated with GRAE are generally more separable compared to those generated by CD, e.g., the classes *Lamp* and *Cap*.

3.5. Ablation Studies

In this experiment, we study how different aspects of GRAE contribute to its overall performance. Particularly, we experiment with two variants of GRAE:

(i) GRAE_F: instead of gradually shrinking the neighborhood size from 10 to 5 during training, this variant fixes the neighborhood size to be $k = 8$; (ii) GRAE_H: this variant only keeps half of the LMD loss during training, and term LMD_rec that computes from the perspective of the reconstructed point cloud is removed.

In general, either fixing the neighborhood size or removing LMD_rec from the complete LMD loss considerably harms the reconstruction performance, in terms of both CD and EMD. Particularly, in GRAE_F, a constant neighborhood size hampers the learning framework to capture the geometry details in a coarse-to-fine manner. Additionally, in GRAE_H, some reconstructed points may not be counted in the loss computation, and thus fail to be improved via backpropagation. By varying the neighborhood as well as fully counting the reconstruction, our proposal, GRAE, evolves into a training paradigm, differentiating it from static loss functions such as CD and EMD.

The reconstruction performance of GRAE and these two variants on the ShapeNet test split and the ModelNet dataset are presented in Table 4 - the lowest loss numbers are in bold. All models are trained only on the ShapeNet training split. As can be seen, the (intact) GRAE almost works consistently better than both GRAE_F and GRAE_H across all architectures in terms of both the CD and EMD metrics. Again, it confirms the effectiveness of gradually shrinking the neighborhood size in GRAE, as well as the usefulness of including the LMD_rec term in the loss computation.

4. REFERENCES

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018. 1, 3

[2] Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. *Advances in neural information processing systems*, 30, 2017. 1

[3] Dimitri P Bertsekas. A new algorithm for the assignment problem. *Mathematical Programming*, 21(1):152–171, 1981. 1

[4] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 1, 3

[5] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, et al. Pot: Python optimal transport. *The Journal of Machine Learning Research*, 22(1):3571–3578, 2021. 1

[6] Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo Rohde. Generalized sliced wasserstein distances. *NeurIPS*, 32, 2019. 3

[7] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 1

[8] Tianyi Lin, Nhat Ho, and Michael I Jordan. On the efficiency of entropic regularized algorithms for optimal transport. *The Journal of Machine Learning Research*, 23(1):6143–6184, 2022. 1

[9] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. Morphing and sampling network for dense point cloud completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34(07), pages 11596–11603, 2020. 1

[10] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In *International Conference on Learning Representations*, 2022. 3

[11] Antonio Montanaro, Diego Valsesia, and Enrico Magli. Rethinking the compositionality of point clouds through regularization in the hyperbolic space. *NeurIPS*, 35:33741–33753, 2022. 3

[12] Jiahao Pang, Duanshun Li, and Dong Tian. Tearingnet: Point cloud autoencoder to learn topology-friendly representations. In *CVPR*, pages 7453–7462, 2021. 3

[13] Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *ICCV*, pages 460–467. IEEE, 2009. 1

[14] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 3, 4

[15] Siyu Ren and Junhui Hou. Unleash the potential of 3d point cloud modeling with a calibrated local geometry-driven distance metric. *arXiv preprint arXiv:2306.00552*, 2023. 1

[16] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 206–215, 2018. 1, 3

[17] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021. 3

[18] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 3d point capsule networks. In *CVPR*, pages 1009–1018, 2019. 3