LEARNING 3D POINT CLOUD EMBEDDINGS USING OPTIMAL TRANSPORT

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning embeddings of any data largely depends on the ability of the target space to capture semantic relations. The widely used Euclidean space, where embeddings are represented as point vectors, is known to be lacking in its potential to exploit complex structures and relations. Contrary to standard Euclidean embeddings, in this work, we embed point clouds as discrete probability distributions in Wasserstein space. We build a contrastive learning setup to learn Wasserstein embeddings that can be used as a pre-training method with or without supervision for any downstream task. We show that the features captured by Wasserstein embeddings are better in preserving the point cloud geometry, including both global and local information, thus resulting in improved quality embeddings. We perform exhaustive experiments and demonstrate the effectiveness of our method for point cloud classification, transfer learning, segmentation and interpolation tasks over multiple datasets including synthetic and real-world objects in both supervised and self-supervised settings. We also compare against other existing methods and show that our method outperforms them in all downstream tasks. Additionally, our study reveals a promising interpretation of capturing critical points of point clouds that makes our proposed method self-explainable.

1 INTRODUCTION

Recent years have seen major advancements in 3D point cloud representation learning. It has gained prominence in a wide spectrum of areas such as robotics (Maturana & Scherer, 2015), computer vision (Su et al., 2015), animation (Pan et al., 2020) with a broad range of applications including shape synthesis and modeling (Yi et al., 2016), autonomous driving (Mahjourian et al., 2018), indoor navigation (Zhu et al., 2017). *Metric learning* for good quality point cloud embeddings is a crucial problem given unique set of challenges associated with 3D data, from processing point clouds in various forms to learning in different spaces. Processing and developing learning methods for point clouds is one of the major challenges due to their irregular, unstructured and unordered nature.

Earlier methods process point clouds by converting them into regular structures like, volumetric representations (Maturana & Scherer, 2015), (Wu et al., 2015) or 2D image projections (Qi et al., 2016), (Su et al., 2015) to employ well explored powerful convolutional techniques. However, these transformations either incur loss of information or require high memory and computational complexity. Later, methods have been developed to learn representations by directly using raw point clouds (Qi et al., 2017a), (Qi et al., 2017b), (Wang et al., 2019). These methods either process each point individually or try to infer features from local regions in a point cloud. The state-of-the-art methods in this category are largely classification, generation or reconstruction-based supervised, unsupervised or self-supervised methods.

The common choice of recent 3D point cloud representation learning methods is to operate and represent point clouds as point vectors in *Euclidean spaces*, where relation between data points is depicted by either angle or distance. We all know that the embedding space largely determines the quality of embeddings, as it depends on how well the *target space* can capture the structure of data. Euclidean space is confined in its potential to capture *complex structure* and possible semantic relations. Realizing these drawbacks, many works use hyperbolic space (Nickel & Kiela, 2018), (Nickel & Kiela, 2017) to capture this uncertainty and asymmetric relationship for word and graph embeddings.

As Euclidean space is constrained in its ability to represent data structures, we need to go *beyond Euclidean space* to get more expressive embeddings for point clouds. Recent studies show that many spaces can be embedded into *Wasserstein space* with low distortion (Frogner et al., 2019), this reflects how large Wasserstein spaces are. Recently, Courty et al. (2018) tries to mimic Wasserstein distance in Euclidean space for image embeddings to build efficient methods along with availing the *flexibility* of Wasserstein space. Also, there are some latest methods for point cloud embeddings using *Optimal Transport (OT) based distances*. Kawano et al. (2020), motivated by Courty et al. (2018), proposes a method to approximate Wasserstein distance by Euclidean norm between two point cloud embeddings. Since Euclidean space is known for its limited ability, finding *isometric low-distortion* point cloud embeddings is tough. Another work by Nguyen et al. (2021) presents how Optimal Transport based distances for point cloud reconstruction affect the quality of learnt embeddings. However, this method utilizes OT based distances only for reconstruction loss, which is not enough to learn *complex shapes* and fails to capture *fine details* of point clouds.

Motivated by aforementioned limitations and inspired by Frogner et al. (2019), in this paper, we advocate for mapping point cloud as a *discrete distribution* in Wasserstein space. We build a contrastive learning setup to learn point cloud embeddings. Leveraging the idea of contrasting point clouds against each other, we intend to learn *common and distinctive features* between same and different distributions, respectively. It can be applied to both *supervised* and *self-supervised* settings. For this, *Sliced Wasserstein (SW) distance* is considered which is a low-cost approximation of Wasserstein distance due to its high computational complexity. Along with comparisons with commonly used distance measures such as L^2 norm and Cosine similarity, we also compare our method against recent works on point clouds using OT. We show that the learnt features capture the point cloud structure better than Euclidean embeddings and consistently performs better in multiple 3D analysis and synthesis tasks. We argue that our approach of incorporating OT metric in a contrastive learning setup captures the underlying geometry and global shape pertaining to critical points (as shown in Figure 1) and fine details of a point cloud.

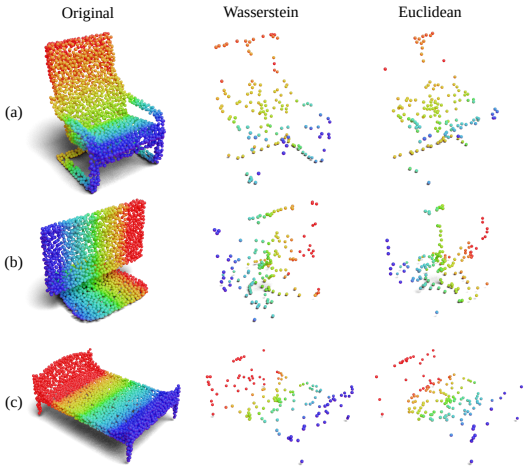

Figure 1: **Critical Points** contribute to the point cloud embedding by capturing global geometry. (a), (b) and (c) represent original point cloud (first column), critical point set for Wasserstein space (second column) and Euclidean space (third column) for three examples (Chair, Monitor & Bed).

Our contributions: i) To the best of our knowledge, we are the first to propose the use of OT metric which exploits the geometry of the data along with contrastive learning for point clouds. Unlike Euclidean embeddings, we represent a point cloud as a discrete probability distribution in the embedding space. ii) Using this representation, we develop a method to learn Wasserstein embeddings for 3D point clouds endowed by contrastive learning setup. We introduce a novel neural network architecture which takes pairs of point clouds as input. It uses supervised/ self-supervised contrastive loss depending on the availability of labels, to minimize the Wasserstein distance between similar point clouds. A major advantage of our network is it can be used as a pretrained model for any downstream network. iii) We perform exhaustive experiments over a wide variety of tasks (supervised and self-supervised learning for classification, transfer learning, segmentation, and interpolation) for four popular point cloud datasets. We show that our Wasserstein embeddings are better in capturing the inherent geometry of point clouds. Additionally, we study the point cloud embeddings in most commonly used Euclidean space for our proposed architecture by replacing the OT metric with L^2 norm (our baseline). We also compare our approach (CL+SW_2) against the other existing methods and show that our method outperforms in all the downstream tasks. iv) We further explore the self-explaining aspect of our model and illustrate the 3D Wasserstein features computed by the encoder (as shown in Figure 1). We show Wasserstein embeddings are better in capturing critical points and semantic structure amenable to the optimization task.

2 PRELIMINARIES

In this section, we briefly present the optimal transport metric, variants of Wasserstein distance, and contrastive learning setup which are used in our proposed method.

2.1 OPTIMAL TRANSPORT AND WASSERSTEIN DISTANCE

Optimal transport aims to solve for the most efficient way to transport mass between two probability distributions. Formally, given two probability distributions μ and ν on a metric space $\mathcal{X}$, for $p \geq 1$, the p-Wasserstein distance is given by

$$W_p(\mu, \nu) = \left(\inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X} \times \mathcal{X}} c(x, y)^p d\pi(x, y) \right)^{1/p} \tag{1}$$

where, π is a transport plan that defines a flow between mass from μ to locations in ν, $\Pi(\mu, \nu)$ is the joint probability distribution with the marginals μ and ν and $c(x, y)$ is the ground metric which assigns a cost of moving a unit of mass $x \in \mathcal{X}$ from μ to some location $y \in \mathcal{X}$ in ν. The cost of moving the mass in μ to match in ν according to the optimal transport plan π^*, is called the Wasserstein distance between the two distributions (Villani, 2003).

The above equation can also be written for discrete distributions, say $\hat{\mu} = \sum_{i=1}^{m} a_i \delta(x_i)$ and $\hat{\nu} = \sum_{j=1}^{n} b_j \delta(y_j)$ are two discrete distributions, where, $\{a_i\}; i = 1 \dots m$ and $\{b_j\}; j = 1 \dots n$ are the probability mass that should sum to 1, δ is the Dirac delta function and $\{x_i\}; i = 1 \dots m$ and $\{y_j\}; j = 1 \dots n$ are the support points in $\mathbb{R}^d$ with m and n being the number of points in each measure. Then, the discrete version of Equation 1 is

$$W_p(\hat{\mu}, \hat{\nu}) = \left(\min_{P \in U(a,b)} \langle C^p, P \rangle \right)^{1/p} \tag{2}$$

where, $\langle \cdot, \cdot \rangle$ denotes the Frobenius dot-product, $C \in \mathbb{R}_+^{m \times n}$ is the pairwise ground metric distance, P is the coupling matrix and U is the set of all possible valid coupling matrices, i.e. $U(a, b) = \{P \in \mathbb{R}^{m \times n} : P\mathbb{1}_n = a, P^\top \mathbb{1}_m = b\}$.

Interestingly, there exists a closed-form solution for Wasserstein distance only when the distributions are one-dimensional measures with L^p norm as the cost function. The closed-form for Wasserstein distance in 1-D is (Peyré & Cuturi, 2019)

$$W_p(\mu, \nu) = \left(\int_0^1 |F_\mu^{-1}(t) - F_\nu^{-1}(t)|^p dt \right)^{1/p} \tag{3}$$

where, F_μ^{-1} and F_ν^{-1} are the inverse cumulative distribution functions of μ and ν.

Generally, we are more interested in dimensions greater than one. Thus, we cannot use this closed-form solution directly to solve the OT problem efficiently. Instead, the Wasserstein distance between two measures on $\mathbb{R}^d$ can be approximated by aggregating the 1-D Wasserstein distance between their projections over multiple directions on a unit sphere, which is called the Sliced Wasserstein distance (Peyré & Cuturi, 2019):

$$SW_p(\mu, \nu) = \left(\int_{S^{d-1}} W_p(P_{\theta,\#}\mu, P_{\theta,\#}\nu)^p d\theta \right)^{1/p} \tag{4}$$

where, $S^{d-1} = \{\theta \in \mathbb{R}^d : \|\theta\| = 1\}$ is the d-dimensional unit sphere and $P_\theta : \mathbb{R}^d \to \mathbb{R}$ is the projection. Since the projections are now 1-D measures, we can use the closed-form solution given by Equation 3. When $m = n$, the Sliced Wasserstein distance can be easily computed by simply sorting points in 1-D measures and can be given by:

$$SW_p(\hat{\mu}, \hat{\nu}) = \left(\frac{1}{D} \sum_{k=1}^{D} \sum_{i=1}^{m} |x_{\alpha_{\theta_k}(i)} - y_{\beta_{\theta_k}(i)}|^p \right)^{1/p} \tag{5}$$

where, α_{θ_k} and β_{θ_k} are the permutation ordering in the increasing order of the support points projected to the direction θ_k with D being the total number of directions.

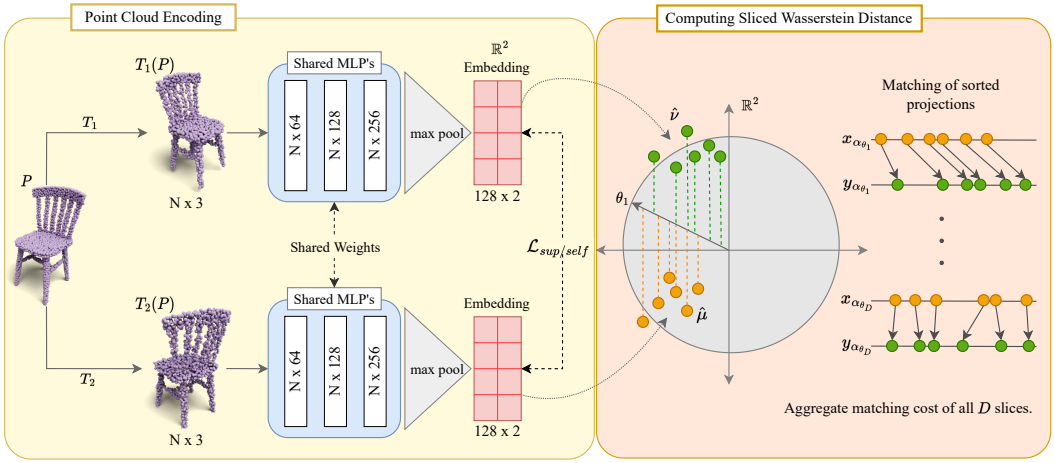

Figure 2: **Overview** of our proposed method. The two main parts are, point cloud encoding as discrete distribution (left) and computation of Sliced Wasserstein distance (right). The ground metric space is $\mathbb{R}^2$. $T_1(P)$ and $T_2(P)$ are two instances of P after random transformations.

2.2 Contrastive Learning

Contrastive learning aims to learn an embedding space that encourages augmentations of the same input sample to have similar representations and of different samples to be dissimilar. Chopra et al. (2005) is an early example of using contrastive learning in a supervised learning setup which takes pair of samples as input to the network.

On the other hand, the contrastive loss introduced by Chen et al. (2020) is named as SimCLR. It follows batch-wise training and is operated in self-supervised setting. For this setup, the distance is reduced between the sample and its augmentations. Later, Khosla et al. (2020) proposed the extension of SimCLR for supervised setup. It additionally aims at reducing the distance between a sample and other samples from same class in a supervised setting.

3 Our Method

In this section, we discuss our method of computing Wasserstein embeddings for point clouds in a contrastive learning setup as shown in Figure 2. We build an in-batch contrastive learning setup which can either be fully supervised or self-supervised and can be used as a pre-training methodology for any downstream task. The goal is to represent samples from same class closer than the samples from different classes in the embeddings space (larger inter-cluster and smaller intra-cluster distance). Here, the choice of embedding space plays a key role for desirable performance, as individual metric spaces can embed data differently and represent different types of semantic structure.

3.1 Contrastive Learning with Optimal Transport

Let $O = \{(P_m, l_m)\}$; $m = 1 \dots M$ be a collection of point clouds $P_m = \{p_i\}$; $i = 1 \dots N_m$, where, $p_i \in \mathbb{R}^3$ with their corresponding class labels $l_m \in L$, where $L = \{1, \dots C\}$ is a set of class labels. Each point cloud P_m contains N_m number of points defined by 3D space points in x, y and z direction. For defining the batch-wise contrastive loss, we first randomly draw K samples from the collection O, that form a batch $B = \{(P_m, l_m)_k\}$; $k = 1 \dots K$. For every point cloud $P_m \in B$, we apply fixed set of random transformations T_1 and T_2 to get two instances of P_m (as shown in Figure 2), giving an augmented batch $B' = \{(P'_m, l_m)_{k'}\}$; $k' = 1 \dots 2K$. The augmented batch is twice the size of the original batch. The point clouds P'_m indexed at k' and $k' + 1$ are augmented version of the point cloud P_m indexed at k. As these are augmented versions of $P_{m[k]}$, their class labels are $l_{m[k']} = l_{m[k'+1]} = l_{m[k]}$.

Table 1: Results of 3D object classification with supervised and self-supervised pre-training on ModelNet10, ModelNet40 and ScanObjectNN (referred as ScanObject) datasets. WPCE and SSW-AE are unsupervised methods and cannot be evaluated for supervised pre-training (represented by "-"). Bold represents the best result and underlined represents the second best.

Method	Supervised Pre-training			Self-Supervised Pre-training		
	ModelNet10	ModelNet40	ScanObject	ModelNet10	ModelNet40	ScanObject
CL+L^2	90.85	84.64	<u>62.82</u>	90.63	84.72	<u>63.51</u>
CL+Cosine	85.90	70.42	56.11	85.90	72.64	56.45
WPCE	-	-	-	89.97	78.84	52.83
SSW-AE	-	-	-	88.88	76.86	51.29
CL+$SW_2(\mathbb{R}^2)$	**91.85**	<u>85.57</u>	61.80	<u>91.41</u>	**85.73**	61.10
CL+$SW_2(\mathbb{R}^4)$	<u>91.74</u>	85.53	61.61	**91.96**	<u>85.45</u>	**63.85**
CL+$SW_2(\mathbb{R}^8)$	91.08	**85.90**	**63.16**	90.19	85.41	60.93

The input to the encoder is an augmented batch B', from which all P'_m needs to be mapped to the embeddings space depending on its geometric features and appearance, with samples having same class label being closer. The encoder represents function $f : \mathbb{R}^{N_m \times 3} \rightarrow \mathcal{W}(\mathcal{X})$, that maps a point cloud P'_m to the Wasserstein space $\mathcal{W}(\mathcal{X})$, with W_p being the distance metric on $\mathcal{W}(\mathcal{X})$ and $\mathcal{X}$ being the ground metric space. We choose $\mathbb{R}^2$, $\mathbb{R}^4$ and $\mathbb{R}^8$ to be our ground metric spaces, in which the corresponding embedding z'_m of P'_m is represented as discrete distribution $\{\frac{1}{S} \cdot x_i\}; i = 1 \dots S$ supported by $x_i \in \mathcal{X}$ with a total of S support points, all with uniform probability mass $\frac{1}{S}$. In our implementation, we reshape the embedding z'_m of P'_m to obtain the discrete distribution for different ground metric spaces.

Generally, the computation for exact solution of W_p is costly. To make the computation of optimal transport more tractable, we replace the distance metric W_p on Wasserstein space $\mathcal{W}(\mathcal{X})$ by the Sliced Wasserstein distance metric SW_p. SW_p is a low-cost approximation of Wasserstein distance with computational complexity being $\mathcal{O}(S \log S)$. For all our experiments, we set the value of $p = 2$ and number of slices $D = 300$.

Supervised Contrastive Loss. In the supervised setting, for any $P'_m \in B'$ indexed at k' with corresponding label $l_{m[k']}$, the positive set is defined as $A = \{P'_m \in B' : P'_m = l_{m[k']}\}$. We define our supervised contrastive loss for learning point cloud Wasserstein embeddings as:

$$\mathcal{L}_{sup} = -\sum_{i=1}^{2K} log \left(\sum_{\substack{j \in A \\ j \neq i}} \frac{exp(-SW_2^2(z_i, z_j))}{\sum_{t \neq i} exp(-SW_2^2(z_i, z_t))} \right) \tag{6}$$

The loss tries to minimize the Sliced Wasserstein distance between the embeddings represented as discrete distribution of an anchor and all the samples having the same class in the augmented batch. This can also be easily converted to a self-supervised version by making necessary modifications.

Self-Supervised Contrastive Loss. Contrary to the supervised setting, in self-supervised setting, the class label of point clouds cannot be used in any way to train the encoder. Here, the positive set of any $P'_m \in B'$ contains only the other augmentation of P'_m. If $i \in \{1 \dots 2K\}$ be the index of any $P'_m \in B'$, then, let $j(i)$ be the index of its other augmented sample. We define our self-supervised loss for learning point cloud Wasserstein embeddings as:

$$\mathcal{L}_{self} = -\sum_{i=1}^{2K} log \left(\frac{exp(-SW_2^2(z_i, z_{j(i)}))}{\sum_{t \neq i} exp(-SW_2^2(z_i, z_t))} \right) \tag{7}$$

Here, only the Sliced Wasserstein distance between embeddings of an anchor and its augmented sample is minimized. Other than the augmented sample, the samples having the same class in the augmented batch are treated as negatives, which might hinder the overall optimization process depending on the batchsize.

4 EXPERIMENTS

Representation that is able to capture good geometric information in a smooth latent space is generally better in various shape understanding and synthesis tasks. To demonstrate the representation power of the learned Wasserstein embeddings compared to Euclidean embeddings, in this section, we present qualitative and quantitative evaluations on multiple tasks: supervised and self-supervised point cloud classification, transfer learning, point cloud segmentation and point cloud interpolation.

Datasets We use ModelNet10 (MN10) and ModelNet40 (MN40) (Wu et al., 2015) to perform experiments on classification. MN40 consists of 12311 CAD models with a total of 40 categories, where 9843 objects are used for training and 2468 for testing. We use the data provided by Qi et al. (2017b), from which we randomly sample 2048 points for each point cloud. MN10 is a subset of MN40 dataset for 10 categories. To evaluate how the learned embeddings perform on real-world data, we also conduct experiments on ScanObjectNN (Uy et al., 2019). It contains object scans with partial occlusions and background making it a challenging dataset. It has 2304 objects for training and 567 for testing from 15 categories. For part segmentation, we use ShapeNetPart (SN) (Yi et al., 2016) that consists of 16681 point clouds from 16 categories and 50 part categories in total.

Pre-training We use a 3-layer MLP followed by a max-pooling layer as our encoder for classification and segmentation tasks. For interpolation, we consider the encoder and decoder proposed by FoldingNet (Yang et al., 2018). In order to perform any downstream task on a particular dataset, the encoder is first pre-trained on the dataset using the contrastive loss explained in Section 3.1 with different distance metrics, followed by testing and evaluation of the desired task. Throughout the experiments, we refer the encoder trained using our method as CL+SW_2 followed by the ground metric space in parenthesis. For the transformations required in contrastive loss, intended towards forming augmented instances, we sequentially compose random scaling, rotation and point jittering. In the case of Euclidean distance metrics, the encoder function $f : \mathbb{R}^{N_m \times 3} \to \mathbb{R}^d$ maps a point cloud to d-dimensional space, that can be interpreted as vectors, with l^2-distance or cosine similarity as distance measures. To account for similarity score given by cosine between two vectors depending on their angles, in Eqs. 6, 7 the negative sign in the numerator should be discarded. Note that when training the encoder with cosine similarity as a distance measure, the embeddings are normalized.

Baselines We consider L^2-distance and Cosine similarity as distance measures for computing Euclidean embeddings. We train the encoder using our loss (Eqs. 6, 7), by replacing $SW_2^2(\cdot, \cdot)$ with these measures in our method. We also consider recent methods for point clouds using Wasserstein metric i.e., WPCE (Kawano et al., 2020) and SSW-AE (Nguyen et al., 2021) as our baselines. WPCE embeds Wasserstein space into Euclidean space using Siamese network. It considers PointNet (Qi et al., 2017a) based encoder-decoder architecture. The network is trained in such a way that the Euclidean distance mimics the Wasserstein distance between two point clouds. SSW-AE proposed to use SW distance and its variants (max SW and adaptive SW) for reconstruction to learn point cloud embeddings. It tries to supervise PointNet based auto-encoder architecture with different metrics.

4.1 3D OBJECT CLASSIFICATION

We extract point cloud embeddings from a pre-trained encoder and use a simple linear SVM as our classifier. Particularly, we fit a linear SVM classifier on the embeddings acquired by an encoder on the train split and report the overall classification accuracy on the test split. In Figure 1, we can see that features captured by Wasserstein embeddings summarize the overall object geometry in a better way compared to the embeddings learned in Euclidean space. This property also reflects in the classification performance shown in Table 1. We can observe that for both supervised and self-supervised settings, the classification accuracy with embeddings extracted by the encoder trained with CL+SW_2 is higher than that of CL+L^2 and CL+Cosine. Thus, compared to Euclidean space, the performance of SW_2 is consistently better on all the datasets, which implies that embeddings learnt in Wasserstein space can increase classification accuracy.

We also show that our method is more effective compared to WPCE and SSW-AE. This improvement can be explained by the difference in the approach of extracting Wasserstein embeddings, where in, our methodology introduces usage of OT metric to directly operate in embedding space endowed by contrastive learning. It helps in learning better representations by exploiting the similarities between distributions along with utilizing the flexibility of the target Wasserstein space.

Table 3: Results of part segmentation with supervised pre-training. Bold represents the best result and underlined represents the second best.

Method	mIoU	aero	bag	cup	car	chair	ear phone	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate board	table
CL+L^2	77.61	73.54	62.76	72.41	64.75	82.42	65.41	88.95	83.01	75.83	93.52	43.03	83.25	74.87	46.01	62.39	77.54
CL+Cosine	74.75	67.80	62.13	**78.66**	66.26	80.00	61.14	86.47	79.34	74.25	92.24	48.70	84.91	70.82	45.63	63.97	73.12
CL+$SW_2(\mathbb{R}^2)$	**81.40**	**80.50**	64.69	74.41	70.97	**87.34**	69.71	89.34	82.96	77.59	**95.31**	**57.28**	88.03	77.14	**53.18**	**69.60**	79.84
CL+$SW_2(\mathbb{R}^4)$	81.03	78.52	61.63	73.82	**71.32**	86.94	**72.48**	90.01	**83.48**	77.78	95.10	52.08	**88.63**	**78.17**	51.41	68.19	79.76
CL+$SW_2(\mathbb{R}^8)$	80.16	78.41	**66.97**	75.67	67.73	84.96	67.74	**90.03**	81.91	**78.06**	94.66	45.95	82.54	75.47	49.40	68.44	**80.01**

Table 4: Results of part segmentation with self-supervised pre-training. Bold represents the best result and underlined represents the second best.

Method	mIoU	aero	bag	cup	car	chair	ear phone	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate board	table
CL+L^2	78.94	75.40	62.74	72.67	67.73	84.73	68.02	89.19	83.35	76.98	94.48	43.15	84.19	75.11	49.60	67.81	77.91
CL+Cosine	78.49	72.51	68.09	71.44	68.35	84.47	63.38	88.69	80.30	75.94	94.45	48.81	88.56	74.08	47.37	69.12	77.90
WPCE	79.92	77.47	**69.06**	74.22	66.59	86.79	66.23	89.30	81.77	75.97	94.60	42.29	**88.71**	74.33	41.05	67.39	79.28
SSW-AE	75.20	68.83	59.61	69.65	64.23	81.98	62.00	86.92	80.27	73.70	92.82	38.46	85.12	68.26	43.97	60.65	73.69
CL+$SW_2(\mathbb{R}^2)$	81.12	78.98	65.46	76.36	**70.36**	**87.45**	68.78	89.45	83.04	77.82	**95.58**	51.21	88.29	**76.61**	**54.42**	70.10	79.71
CL+$SW_2(\mathbb{R}^4)$	**81.17**	78.98	66.90	**77.98**	70.35	86.91	**70.57**	89.39	82.85	**77.99**	94.77	**56.20**	87.18	76.10	53.98	69.60	80.10
CL+$SW_2(\mathbb{R}^8)$	80.97	**79.69**	64.27	77.42	69.64	86.54	67.35	**89.72**	**83.59**	77.22	94.47	48.35	88.45	74.67	50.99	**70.42**	**80.33**

4.2 TRANSFER LEARNING

We examine the generalizing ability of the embeddings acquired by encoders trained with different distance metrics to unseen classes, by performing transfer learning for point cloud classification. We follow the same process as explained in Section 4.1 for reporting the overall classification accuracy. The quantitative comparisons of transfer learning is shown in Table 2. We perform evaluation in two transfer learning settings, MN10 to MN40 and SN to MN40. Here, the encoder is pre-trained on MN10 and SN followed by evaluation on MN40. In both the settings, the model generalizes to new unseen classes by wielding the knowledge of geometry learned during training. We can see that CL+SW_2 consistently performs better than other distance measures and methods in both the transfer learning settings with and without supervision. Results imply that Wasserstein embeddings are better in transferring the knowledge of capturing geometry for yielding good classification performance.

Table 2: Results of 3D object classification with self-supervised pre-training for *transfer learning* setup. WPCE and SSW-AE are unsupervised methods and cannot be evaluated for supervised pre-training (represented by "-"). Bold represents the best result and underlined represents the second best.

Method	Supervised Pre-training		Self-Supervised Pre-training	
	MN10 to MN40	SN to MN40	MN10 to MN40	SN to MN40
CL+L^2	85.37	85.81	84.27	83.83
CL+Cosine	74.51	69.12	75.32	71.47
WPCE	-	-	77.51	78.03
SSW-AE	-	-	76.05	76.66
CL+$SW_2(\mathbb{R}^2)$	85.70	85.61	**85.61**	**85.77**
CL+$SW_2(\mathbb{R}^4)$	**86.18**	**86.18**	85.57	84.88
CL+$SW_2(\mathbb{R}^8)$	85.57	85.57	85.49	85.21

4.3 3D OBJECT PART SEGMENTATION

We train a 3-layer MLP network to predict a class label for all points in a point cloud, where the input to this network is the embedding provided by a pre-trained encoder. In particular, part segmentation requires fine-grain understanding of the local geometry of the objects. Along with the global embedding of the point cloud, per point embeddings acquired before max-pooling are stacked together and passed to the segmentation network. Note that, only the segmentation network weights are optimized, using the standard cross-entropy loss and the encoder's weights are frozen. We evaluate the performance using mIoU metric. For mIoU of each class, the IoU's of all parts from that class are averaged. Instance average mIoU is calculated by taking the mean of IoU's for all the instances. The comparison of average instance mIoU and per class average mIoU for both supervised and self-supervised learning settings are shown in Table 3 and Table 4, respectively. We can see that

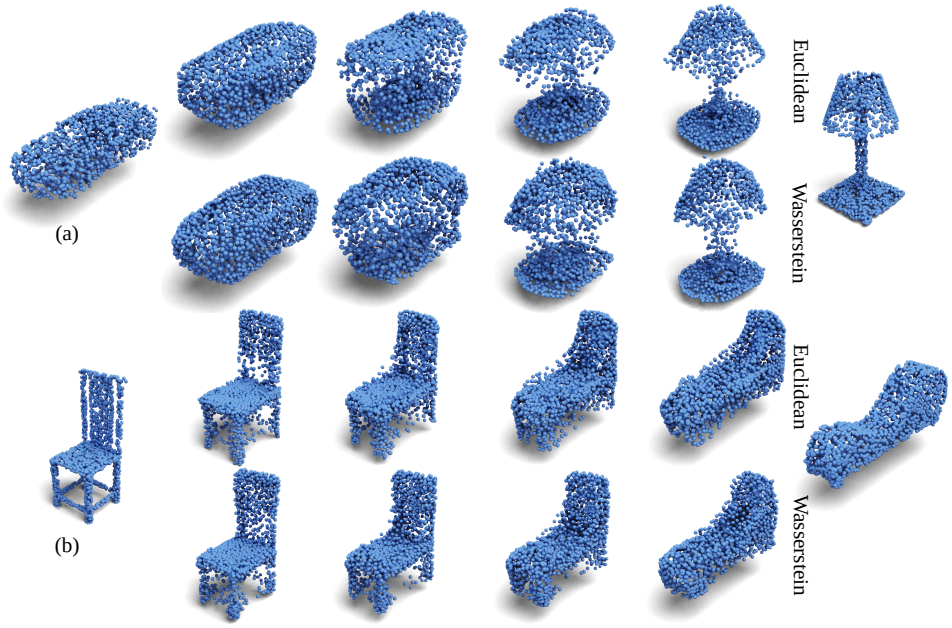

Figure 3: **Interpolation** (linear-combination) between source and target for two examples (a) Car to Lamp, (b) Chair to Chair from ShapeNet. The left and the right most represent original point clouds. Top row of each example show results from reconstruction after interpolating two point cloud embeddings in Euclidean space. The bottom row of each example provide interpolation results in Wasserstein space. All rows follow the ratio of 0.2, 0.4, 0.6 and 0.8 (from left to right).

the results outperform other distance measures and methods, implying that Wasserstein embeddings are able to capture better fine-grain local information required for the task.

4.4 3D Shape Interpolation

We further examine the quality of our learnt space by performing shape interpolation between inter and intra class point cloud instances. The main aim of conducting this task is to examine which learnt space is capable of capturing geometric information needed to generate consistent interpolations of 3D point clouds based on their structure. As interpolation is a synthesis task, we need a decoder network to reconstruct the object from its embedding. For this, we train an encoder-decoder network with our contrastive loss (Eq. 6) on the embeddings for the encoder, along with a reconstruction loss for the decoder. We use the encoder and decoder proposed by FoldingNet, which learns to deform a unit sphere and take the shape of a 3D object's surface. We found that optimizing the network for better classification performance, as well as getting detailed reconstruction is difficult. As our contrastive loss aims to pull point clouds closer with similar global representations, it becomes difficult to accurately reconstruct the input point cloud without fine-grain characteristic information. A simple way to deal with this issue is to assign weightage to the individual loss terms, with the weights summing to 1. In order to train an encoder-decoder, the total effective loss is defined by taking a weighted sum of our contrastive loss and a reconstruction loss, with weights being 0.2 and 0.8, respectively. We use Chamfer distance as the reconstruction loss. Interpolation results are shown in Figure 3. We can see that the interpolations done using Wasserstein embeddings follow a smooth path with relatively less noisy points. For example, in Figure 3 (b), we can see that for Euclidean, the source chair suddenly transforms to take the shape of target chair, whereas in Wasserstein, the legs of chair smoothly morph to become the base of target chair.

4.5 Explainability

We investigate what makes Wasserstein embeddings perform better as shown in the downstream tasks. We visualize and compare the features captured by Wasserstein embeddings and Euclidean

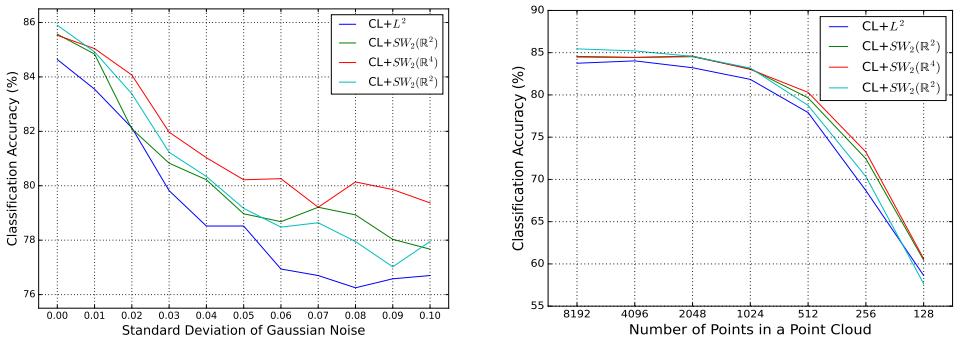

Figure 4: **Ablation Study:** Classification accuracy for noise model with Gaussian Noise (left) and points removal using random sampling (right) for our method CL+SW_2 and our baseline CL+L^2.

embeddings in Figure 1. These features are called critical points, as shown by Qi et al. (2017a). The embedding of a point cloud is completely determined by these subset of points. The embedding for a point cloud would be the same, as long as, the set of critical points is unchanged. For a given point cloud, the critical points are those 3D points that contribute to the global embeddings after the max pooling layer. This implies that the number of critical points cannot be greater than that of the embedding size. The selection of critical points is extremely important, as they solely decide the embedding of a point cloud. This makes it clear that for good quality embeddings, critical points should best describe the given point cloud. In Figure 1, we can see that the network intelligently tries to summarize the point cloud by choosing boundary points as the critical points. Our Wasserstein embeddings are able to capture the full skeleton structure of the given point cloud, whereas, critical points captured by Euclidean embeddings are comparatively poor with uneven distribution and missing parts. Thus, we can say that Wasserstein space are indeed better in preserving and capturing geometric structure amenable to the optimization task.

4.6 Ablation Study

We perform point perturbation and point density variation to test their effects on the encoders pretrained with different distance metrics and report the classification accuracy on Modelnet40 as shown in Figure 4. For the point perturbation test, we add Gaussian noise to input point clouds, with standard-deviation of noise varying from 0.01 to 0.1. We can observe that for all noise levels, even with severe distortion, CL+SW_2 performs well than that of CL+L^2. This implies that discrete representation learnt in Wasserstein space is less prone to performance degradation due to noise in inputs. Further, for varying density test, we randomly sample 8192, 4096, 2048, 1024, 512, 256, 128 points from input point clouds and perform evaluation on them. We can observe that CL+SW_2 consistently does better than CL+L^2. This shows Wasserstein embeddings are robust towards missing points in the input point cloud.

5 Conclusion

In this paper, we proposed to represent point clouds as discrete probability distributions in the Wasserstein space. We built a contrastive learning method to learn Wasserstein embeddings for 3D point clouds. Our proposed method can be used as a pretrained model for any downstream network in supervised and self-supervised settings. Empirically, we found that representations learnt using our pre-training of contrastive learning with Sliced Wasserstein distance captured the structure and underlying geometry better than standard Euclidean embeddings. With improved embeddings, our method outperformed all the existing methods including our baseline with L^2 norm and Cosine similarity for all the downstream tasks (classification, segmentation, transfer learning, interpolation) in both supervised and self-supervised settings. We also show an interesting study of our self-explainable method by capturing critical points of point clouds better than embeddings in Euclidean space. For future work, a possible direction is to explore other related problems such as domain adaptation for point clouds using optimal transport. Another interesting aspect is to consider complex datasets including multiple objects and scenes of point clouds.

Reproducibility Statement:
Our proposed method is easily reproducible considering pairs of point clouds as input. The network architecture explained in Figure 2 consists of simple MLP layers followed by max pooling and reshaping of embedding. We mention contrastive loss functions for supervised and self-supervised settings in Section 3.1. The pre-training setup is detailed in Experiments Section 4. Datasets used in this paper are well-known in point cloud domain. We provided references for all the datasets in Experiments Section. Our code will be made publicly available after the acceptance of the work.

REFERENCES

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning (ICML)*, pp. 1597–1607. PMLR, 2020.

S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 539–546, 2005.

Nicolas Courty, Rémi Flamary, and Mélanie Ducoffe. Learning wasserstein embeddings. In *International Conference on Learning Representations (ICLR)*, pp. 1–13, 2018.

Charlie Frogner, Farzaneh Mirzazadeh, and Justin Solomon. Learning entropic wasserstein embeddings. In *International Conference on Learning Representations (ICLR)*, 2019.

Keisuke Kawano, Satoshi Koide, and Takuro Kutsuna. Learning wasserstein isometric embedding for point clouds. In *International Conference on 3D Vision (3DV)*, pp. 473–482, 2020.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 18661–18673, 2020.

Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 5667–5675, 2018.

Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928, 2015.

Trung Nguyen, Quang-Hieu Pham, Tam Le, Tung Pham, Nhat Ho, and Binh-Son Hua. Point-set distances for learning representations of 3D point clouds. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 6341–6350, 2017.

Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning (ICML)*, pp. 3776–3785. PMLR, 2018.

Linfei Pan, Ľubor Ladický, and Marc Pollefeys. Compression and completion of animated point clouds using topological properties of the manifold. In *International Conference on 3D Vision (3DV)*, pp. 734–742, 2020.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, pp. 355–602, 2019.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017a.

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 5105–5114, 2017b.

Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas Guibas. Volumetric and multi-view CNNs for object classification on 3D data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 945–953, 2015.

Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)*, 2019.

Villani. *Topics in Optimal Transportation*. American Mathematical Society, 2003.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

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)*, pp. 1912–1920, 2015.

Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. FoldingNet: Point cloud auto-encoder via deep grid deformation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 206–215, 2018.

Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics (TOG)*, 2016.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *IEEE international conference on robotics and automation (ICRA)*, pp. 3357–3364, 2017.