Point-set Distances for Learning Representations of 3D Point Clouds

Trung Nguyen[1] Quang-Hieu Pham[3] Tam Le[4] Tung Pham[1] Nhat Ho[5] Binh-Son Hua[1,2]
[1]VinAI Research, Vietnam [2]VinUniversity, Vietnam
[3]Woven Planet North America, Level 5 [4]RIKEN AIP, Japan [5]University of Texas, Austin

Abstract

Learning an effective representation of 3D point clouds requires a good metric to measure the discrepancy between two 3D point sets, which is non-trivial due to their irregularity. Most of the previous works resort to using the Chamfer discrepancy or Earth Mover's distance, but those metrics are either ineffective in measuring the differences between point clouds or computationally expensive. In this paper, we conduct a systematic study with extensive experiments on distance metrics for 3D point clouds. From this study, we propose to use sliced Wasserstein distance and its variants for learning representations of 3D point clouds. In addition, we introduce a new algorithm to estimate sliced Wasserstein distance that guarantees that the estimated value is close enough to the true one. Experiments show that the sliced Wasserstein distance and its variants allow the neural network to learn a more efficient representation compared to the Chamfer discrepancy. We demonstrate the efficiency of the sliced Wasserstein metric and its variants on several tasks in 3D computer vision including training a point cloud autoencoder, generative modeling, transfer learning, and point cloud registration.

1. Introduction

Since the spark of the modern artificial intelligence, 3D deep learning on point clouds has become a powerful technique for solving recognition problems such as object classification [44, 21], object detection [43], and semantic segmentation [41]. Generative modeling with 3D point clouds has also been studied with some promising results [51, 46, 32, 22, 47, 33]. Another 3D computer vision problem that has seen the rise of deep learning approaches is point cloud matching [8, 11, 10, 17]. All of these problems share a common task — that is to learn a robust representation of 3D point clouds.

One of the most important steps in learning representations of 3D point clouds is to choose a metric to measure the discrepancy between two point sets. There are two popular choices for such metric: the Chamfer divergence and the Earth Mover's distance (EMD) [16]. While earlier works [16, 1] has shown that EMD performs better than Chamfer in terms of learning representations, Chamfer is more favored [10, 52, 18, 15, 12, 19] due to its significantly lower computational cost.

In this article, we revisit the similarity metric problem in 3D point cloud deep learning. We propose to use the sliced Wasserstein distance (SWD) [5], which is based on projecting the points in point clouds into a line, and its variants as effective metrics to supervise 3D point cloud autoencoders. Compared to Chamfer divergence, SWD is more suitable for point cloud reconstruction, while remaining computationally efficient (cf. Figure 1). We show that Chamfer divergence is weaker than the EMD and sliced Wasserstein distance (cf. Lemma 1) while the EMD and sliced Wasserstein distance are equivalent. It suggests that even when two point clouds are close in Chamfer divergence, they may not be close in either the EMD or sliced Wasserstein distance. Furthermore, the sliced Wasserstein distance has a computational complexity in the order of $N \log N$ [5], which is comparable to that of the Chamfer divergence, while EMD has a complexity in the order of N^3 [39] where N is the number of points in 3D point clouds. Finally, under the standard point clouds settings, since the dimension of points is usually three, the projection step in sliced Wasserstein distance will only lead to small loss of information of the original point clouds. As a consequence, the sliced Wasserstein distance possesses both the computational and statistical advantages for point cloud learning over Chamfer divergence and EMD. To improve the quality of slices from SWD, we also discuss variants of sliced Wasserstein distance, including max-sliced Wasserstein distance [13] and the proposed adaptive sliced Wasserstein algorithm.By conducting a case study on learning a 3D point cloud auto-encoder, we provide a comprehensive benchmark on the performance of different metrics. These results align with our theoretical development. In summary, our main findings are:

- A first theoretical study about the relation between Chamfer divergence, EMD, and sliced Wasserstein distance for point cloud learning (Section 4).

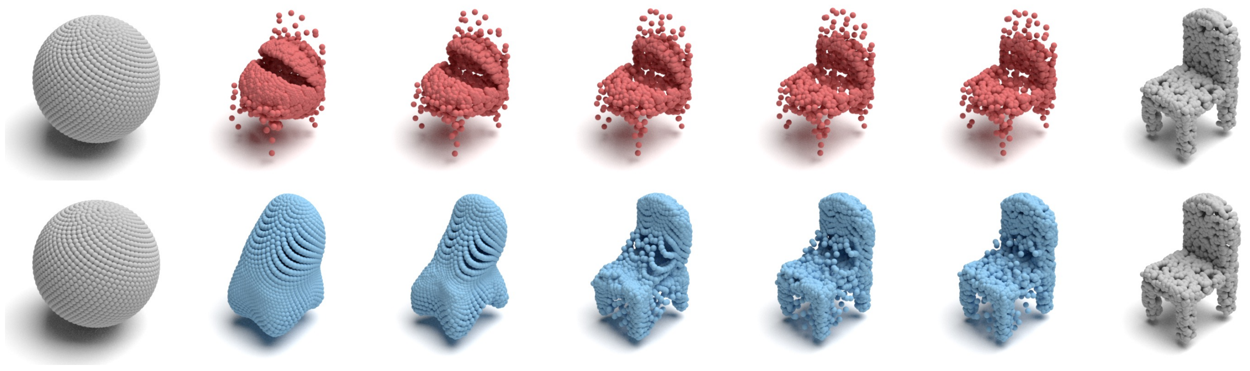

Figure 1: We advocate the use of sliced Wasserstein distance for training 3D point cloud autoencoders. In this example, we try to morph a sphere into a chair by optimizing two different loss functions: Chamfer discrepancy (top, red) and sliced Wasserstein distance (bottom, blue). The proposed sliced Wasserstein distance only takes 1000 iterations to converge, while it takes 50000 iterations for Chamfer discrepancy.

- A new algorithm named Adaptive-sliced Wasserstein to evaluate sliced Wasserstein distance that guarantees that the evaluated value is close enough to the true one (Section 4).
- An extensive evaluation of point cloud learning tasks including point cloud reconstruction, transfer learning, point cloud registration and generation based on Chamfer discrepancy, EMD, sliced Wasserstein distance and its variants (Section 5).

2. Related Work

Set similarity. 3D point clouds autoencoders are useful in a wide range of applications, such as denoising [19], 3D matching [55, 10, 12, 31], and generative models [16, 1]. Many autoencoder architectures have been proposed in recent few years [1, 52, 10]. To train these autoencoders, there are two popular choices of losses: the Chamfer discrepancy (CD) and Earth Mover's distance (EMD). The Chamfer discrepancy has been widely used in point cloud deep learning [16, 1, 52].

It is known that Chamfer discrepancy (CD) is not a distance that means there are two different point clouds with its CD almost equals zero. While earlier works [16, 1] showed that EMD is better than Chamfer in 3D point clouds reconstruction task, recent works [42] still favor Chamfer discrepancy due to its fast computation.

Wasserstein distance. In 2D computer vision, the family of Wasserstein distances and their sliced-based versions have been considered in the previous works [2, 20, 48, 14, 13, 49, 29, 23]. In particular, Arjovsky et al. [2] proposed using Wasserstein as the loss function in generative adversarial networks (GANs) while Tolstikhin et al. [48] proposed using that distance for the autoencoder framework. Nevertheless, Wasserstein distances, including the EMD, have expensive computational cost and can suffer from the curse of dimensionality, namely, the number of data required to train the model will grow exponentially with the dimension. To deal with these issues of Wasserstein distances, a line of works has utilized the slicing approach to reduce the dimension of the target probability measures. The notable slicing distance is sliced Wasserstein distance [5]. Later, the idea of sliced Wasserstein distance had been adapted to the autoencoder setting [25] and domain adaptation [30]. Deshpande et al. [14] proposed to use the max-sliced Wasserstein distance, a version of sliced Wasserstein distance when we only choose the best direction to project the probability measures, to formulate the training loss for a generative adversarial network. The follow-up work [13, 49] has an improved projection complexity compared to the sliced Wasserstein distance. In the recent work, Nguyen et al. [37, 38] proposed a probabilistic approach to chooses a number of important directions via finding an optimal measure over the projections. Another direction with sliced-based distances is by replacing the linear projections with non-linear projections to capture more complex geometric structures of the probability measures [24]. However, to the best of our knowledge, none of such works have considered the problem of learning with 3D point clouds yet.

Notation. Let $\mathbb{S}^{n-1}$ be the unit sphere in the n-dimensional space. For a metric space (Ω, d) and two probability measures μ and ν on Ω, let $\Pi(\mu, \nu)$ be the set of all joint distributions γ such that its marginal distributions are μ and ν, respectively. For any $\theta \in \mathbb{S}^{n-1}$ and any measure μ, $\pi_\theta \sharp \mu$ denotes the pushforward measure of μ through the mapping $\mathcal{R}_\theta$ where $\mathcal{R}_\theta(x) = \theta^\top x$ for all x.

3. Background

To study the performance of different metrics for point cloud learning, we briefly review the mathematical foundation of the Chamfer discrepancy and the Wasserstein distance, which serve as the key building blocks in this paper. Note that in computer vision, Chamfer is often abused to be a distance. Strictly speaking, Chamfer is a pseudo-distance, *not* a distance [16]. Therefore, in this paper, we use the terms Chamfer discrepancy or Chamfer divergence instead.

3.1. Chamfer discrepancy

In point cloud deep learning, Chamfer discrepancy has been adopted for many tasks. There are some variants of Chamfer discrepancy, which we provide here for completeness. For any two point clouds P, Q, a common formulation of the Chamfer discrepancy between P and Q is given by:

$$d_{\text{CD}}(P,Q) = \frac{1}{|P|}\sum_{x\in P}\min_{y\in Q}||x-y||_2^2 + \frac{1}{|Q|}\sum_{y\in Q}\min_{x\in P}||x-y||_2^2. \tag{1}$$

A slightly modified version of Chamfer divergence is also used by previous works [52, 10, 12, 3] that replaces the sum by a max function:

$$d_{\text{MCD}}(P,Q) = \max\Big\{\frac{1}{|P|}\sum_{x\in P}\min_{y\in Q}||x-y||_2^2, \frac{1}{|Q|}\sum_{y\in Q}\min_{x\in P}||x-y||_2^2\Big\}. \tag{2}$$

In both definitions, the min function means that Chamfer discrepancy only cares about the nearest neighbour of a point rather than the distribution of those nearest points. Hence, as long as the supports of x and y are close then the corresponding Chamfer discrepancy between them is small, meanwhile their corresponding distributions could be different. A similar phenomenon, named Chamfer blindness, was shown in [1], pointing out that Chamfer discrepancy fails to distinguish bad sample from the true one, since it is less discriminative.

3.2. Wasserstein distance

Let (Ω, d) be a metric space, μ, ν are probability measures on Ω. For $p \geq 1$, the p-Wasserstein distance (WD) is given by

$$W_p(\mu,\nu) = \inf_{\gamma\in\Pi(\mu,\nu)}\Big\{\mathbb{E}_{(X,Y)\sim\gamma}\big[d^p(X,Y)\big]\Big\}^{\frac{1}{p}}.$$

For $p = 1$, the Wasserstein distance becomes the Earth Mover's Distance (EMD), where the optimal joint distribution γ induces a map $T : \mu \to \nu$, which preserves the measure on any measurable set $B \subset \Omega$. In the one-dimensional case $\Omega = \mathbb{R}$ and $d(x,y) := |x-y|$, the WD has the following closed-form formula:

$$W_p(\mu,\nu) = \Big(\int_0^1 \big|F_X^{-1}(t) - F_Y^{-1}(t)\big|^p, dt\Big)^{\frac{1}{p}}, \tag{3}$$

where F_X and F_Y are respectively the cumulative distribution functions of random variables X and Y. When the dimension is greater than one, there is no closed-form for the WD, that makes calculating the WD more difficult.

EMD in 3D point-cloud applications. For the specific settings of 3D point clouds, the EMD had also been employed to define a metric between two point clouds [16, 1, 53]. With an abuse of notation, for any two given point clouds P and Q, throughout this paper, we denote its measure representation as follows: $P = \frac{1}{|P|}\sum_{x\in P}\delta_x$ and $Q = \frac{1}{|Q|}\sum_{y\in Q}\delta_y$ where δ_x denotes the Dirac delta distribution at point x in the point cloud P. When $|P| = |Q|$, the Earth Mover's distance [16, 1, 53] between P and Q is defined as

$$d_{\text{EMD}}(P,Q) = \min_{T:P\to Q}\sum_{x\in P}||x - T(x)||_2. \tag{4}$$

While earlier works [16, 1] showed that EMD is better than Chamfer in 3D point clouds reconstruction task, the computation of EMD can be very expensive compared to the Chamfer divergence. In particular, it had been shown that the practical computational efficiency of EMD is at the order of $\max\{|P|, |Q|\}^3$ [39], which can be expensive. There is a recent line of work using entropic version of EMD or in general Wasserstein distances [9] to speed up the computation of EMD. However, the best known practical complexity of using the entropic approach for approximating the EMD is at the order $\max\{|P|, |Q|\}^2$ [34, 35], which is still expensive and slower than that of Chamfer divergence. Therefore, it necessitates to develop a metric between 3D point clouds such that it not only has equivalent statistical properties as those of EMD but also has favorable computational complexity similar to that of the Chamfer divergence.

4. Sliced Wasserstein Distance and its variants

In this section, we first show that Chamfer divergence is a weaker divergence than Earth Mover's distance in Section 4.1. Since Earth Mover's distance can be expensive to compute, we propose using sliced Wasserstein distance, which is equivalent to Wasserstein distance and has efficient computation, as an alternative to EMD in Section 4.2.1. Finally, we propose a new algorithm to compute sliced-Wasserstein that can guarantee the estimated value is close enough to the true one in Section 4.2.2.

4.1. Relation between Chamfer divergence and Earth Mover's distance

In this section, we study the relation between Chamfer and EMD when $|P| = |Q|$. In particular, the following inequality shows that the Chamfer divergence is weaker than the Wasserstein distance.

Lemma 1. *Assume $|P| = |Q|$ and the support of P and Q is bounded in a convex hull of diameter K, then we find that*

$$d_{CD}(P,Q) \le 2K d_{EMD}(P,Q). \quad (5)$$

Proof. Assume that T is the optimal plan from P to Q. Then, we find that

$$\min_{y\in Q} \|x-y\|_2 \le \|x-T(x)\|_2$$
$$\Rightarrow \min_{y\in Q} \|x-y\|_2^2 \le K\|x-T(x)\|_2,$$

since $\|x-T(x)\|_2 \le K$. Taking the sum over x, we obtain

$$\sum_{x\in P} \min_{y\in Q} \|x-y\|_2^2 \le K \sum_{x\in P} \|x-T(x)\|_2.$$

Similarly we have $\sum_{y\in Q} \min_{x\in P} \|x-y\|_2^2 \le K\sum_{y\in Q}\|y-\bar{T}(y)\|_2$ where $\bar{T}$ is the optimal plan from Q to P. Then we obtain the desired inequality. □

The inequality in Lemma 1 implies that minimizing the Wasserstein distance leads to a smaller Chamfer discrepancy, and the reverse inequality is not true. Therefore, EMD and Chamfer divergence are not equivalent, which can be undesirable. Note that the inequality could not be improved to a better order of K. For example, lets consider two point-clouds that have small variance ϵ^2, meanwhile the distance between two point-cloud centers equals $K - O(\epsilon)$. Then the CD is of order K^2, the EMD is of order K.

Lemma 1 shows that Chamfer divergence is weaker than the EMD, which is in turn weaker than other divergences such as, KL, chi-squared, etc. [40, pp.117]. However, Chamfer discrepancy has weakness as we explained before, and other divergences are not as effective as EMD, since they are very loose even when two distributions are close to each other [2]. Hence, the Wasserstein/EMD is a preferable metric for learning the discrepancy between two distributions.

Despite that fact, computing EMD can be quite expensive since it is equivalent to solving a linear programming problem, which has the best practical computational complexity of the order $\mathcal{O}(\max\{|P|^3, |Q|^3\})$ [39]. On the other hand, the computational complexity of Chamfer divergence can be scaled up to the order $\mathcal{O}(\max\{|P|, |Q|\})$. Therefore, Chamfer divergence is still preferred in practice due to its favorable computational complexity.

Given the above observation, ideally, we would like to utilize a distance between P and Q such that it is equivalent to the EMD and has linear computational complexity in terms of $\max\{|P|, |Q|\}$, which is comparable to the Chamfer divergence. It leads us to the notion of sliced Wasserstein distance in the next section.

4.2. Sliced Wasserstein distance and its variants

In order to circumvent the high computational complexity of EMD, the sliced Wasserstein distance [5] is designed to exploit the 1D formulation of Wasserstein distance in Equation (3).

4.2.1 Sliced Wasserstein distance

In particular, the idea of sliced Wasserstein distance is that we first project both target probability measures μ and ν on a direction, says θ, on the unit sphere to obtain two projected measures denoted by $\pi_\theta \sharp \mu$ and $\pi_\theta \sharp \nu$, respectively. Then, we compute the Wasserstein distance between two projected measures $\pi_\theta \sharp \mu$ and $\pi_\theta \sharp \nu$. The sliced Wasserstein distance (SWD) is defined by taking the average of the Wasserstein distance between the two projected measures over all possible projected direction θ. In particular, for any given $p \ge 1$, the sliced Wasserstein distance of order p is formulated as follows:

$$SW_p(\mu,\nu) = \Big(\int_{\mathbb{S}^{n-1}} W_p^p\big(\pi_\theta \sharp \mu, \pi_\theta \sharp \nu\big) d\theta\Big)^{\frac{1}{p}}. \quad (6)$$

The SW_p is considered as a low-cost approximation for Wasserstein distance as its computational complexity is of the order $\mathcal{O}(n \log n)$ where n is the maximum number of supports of the discrete probability measures μ and ν. When $p = 1$, the SW_p is weakly equivalent to first order WD or equivalently EMD [4]. The equivalence between SW_1 and EMD along with the result of Lemma 1 suggests that SW_1 is stronger metric than the Chamfer divergence while it has an appealing optimal computational complexity that is linear on the number of points of point clouds, which is comparable to that of Chamfer divergence.

We would like to remark that since the dimension of points in point clouds is generally small (≤ 6), sliced Wasserstein distance will still be able to retain useful information of the point clouds even after we project them to some direction on the sphere. Due to its favorable computational complexity, sliced Wasserstein distance had been used in several applications: point cloud registration [28], generative models on 2D images; see, for examples [45, 36, 14, 25, 26, 49, 27]. However, to the best of our knowledge, this distance has not been used for deep learning tasks on 3D point clouds.

Monte Carlo estimation. In Equation (6), the integral is generally intractable to compute. Therefore, we need to approximate the integral. A common approach to approximate the integral is by applying the Monte Carlo method. In particular, we sample N directions $\theta_1, \ldots, \theta_N$ uniformly from the sphere $\mathbb{S}^{\mathrm{d}-1}$ where d is the dimension of points in point clouds, which results in the following approximation of sliced Wasserstein distance:

$$SW_p(\mu,\nu) \approx \left(\frac{1}{N}\sum_{i=1}^{N} W_p^p\big(\pi_{\theta_i}\sharp\mu, \pi_{\theta_i}\sharp\nu\big)\right)^{\frac{1}{p}}. \quad (7)$$

where the number of slices N is tuned for the best performance.

Since N plays a key role in determining the approximation of sliced Wasserstein distance, it is usually chosen based on the dimension of the probability measures μ and ν. In our applications with 3D point clouds, since the dimension of the points in point clouds is generally small, we observe that choosing the number of projections N up to 100 is already sufficient for learning 3D point clouds well.

Max-sliced Wasserstein distance. To avoid using uninformative slices in SWD, another approach focuses on taking only the best slice in discriminating between two given distributions. That results in max-sliced Wasserstein distance [13]. For any $p \geq 1$, the *max-sliced Wasserstein distance* of order p is given by:

$$MSW_p(\mu,\nu) := \max_{\theta\in\mathbb{S}^{n-1}} W_p\big(\pi_\theta\sharp\mu, \pi_\theta\sharp\nu\big). \quad (8)$$

4.2.2 An adaptive-sliced Wasserstein algorithm

Another drawback of the Monte Carlo estimation in SWD is that it does not give information about how close the estimated value is to the true one. Therefore, we introduce the novel *adaptive-sliced Wasserstein algorithm* (ASW). In particular, given N uniform random projections $\{\theta_i\}_1^N$ drawn from the sphere $\mathbb{S}^{n-1}$, for the simplicity of presentation, we denote $sw_i = W_p^p(\pi_{\theta_i}\sharp\mu, \pi_{\theta_i}\sharp\nu)$ for all $1 \leq i \leq N$. Furthermore, we denote $\overline{sw} = SW_p^p(\mu,\nu)$ the true value of SW distance that we want to compute. The Monte Carlo estimation of the SW distance can be written as follows: $\overline{sw}_N := \frac{1}{N}\sum_{i=1}^N sw_i$, which is an unbiased estimator of the true value $\overline{sw}$. Similarly, the biased and unbiased variance estimates are respectively defined as:

$$s_N^2 := \frac{1}{N}\sum_{i=1}^{N}\big(sw_i - \overline{sw}_N\big)^2, \quad \bar{s}_N^2 := \frac{N}{N-1}s_N^2. \quad (9)$$

Our idea of adaptivity is to dynamically determine the number of projections N from the observed mean and variance of the estimators. To do so, we leverage a probabilistic bound of the error of the estimator and choose N such that the error bound is below a certain tolerance threshold. Particularly, applying central limit theorem, we have

$$\mathbb{P}\left(|\overline{sw}_N - \overline{sw}| < \frac{k\bar{s}_N}{\sqrt{N}}\right) \approx \phi(k) - \phi(-k) \quad (10)$$

where $\phi(k) := \int_{-\infty}^{k} \frac{e^{-x^2/2}}{\sqrt{2\pi}}dx$. We set $k = 2$ so that the above probability is around 95%.

Algorithm 1: Adaptive sliced Wasserstein.

```
Input: Two point sets, positive integers $N_0, s$; $\epsilon > 0$;
  maximum number of projections $M$
Ouput: $\overline{sw}_N$
Sample $N_0$ projections ;
Compute $\overline{sw} := \overline{sw}_{N_0}, \overline{sw^2} := \overline{sw^2}_{N_0}, N := N_0$ ;
while $\overline{sw^2} - (\overline{sw})^2 > \frac{(N-1)\epsilon^2}{4}$ & $N \leq M$ do
    Sample $s$ projections ;
    Compute $\overline{sw}_s, \overline{sw^2}_s$ ;
    Assign $\overline{sw} := \frac{N\times\overline{sw}+s\times\overline{sw}_s}{N+s}$ ;
    Assign $\overline{sw^2} := \frac{N\times\overline{sw^2}+s\times\overline{sw^2}_s}{N+s}$ ;
    Assign $N := N + s$ ;
end
```

Given a predefined tolerance $\epsilon > 0$, we aim for

$$\frac{k\bar{s}_N}{\sqrt{N}} \leq \epsilon, \text{ or } \frac{k^2\bar{s}_N^2}{N} \leq \epsilon^2. \quad (11)$$

From Equation (9), we note that $\frac{\bar{s}_N^2}{N} = \frac{s_N^2}{N-1}$ and so it is desirable to choose N such that $\frac{k^2 s_N^2}{N-1} \leq \epsilon^2$. Rewriting the biased variance in Equation (9), we get $s_N^2 = \frac{1}{N}\sum_{i=1}^N sw_i^2 - (\overline{sw}_N)^2$. Denote $\overline{sw^2}_N := \frac{1}{N}\sum_{i=1}^N sw_i^2$, the condition becomes

$$\overline{sw^2}_N - (\overline{sw}_N)^2 \leq \frac{(N-1)\epsilon^2}{k^2}.$$

That leads us to the construction of Algorithm 1. In this algorithm, we start by estimating the SWD with an initial number of projections $N = N_0$, and then dynamically update N with extra s samples by estimating the online mean and variance of the distance estimator until the estimated error satisfies the error bound. We note that ASW algorithm can be used to compute other variants of SWD, such as, generalized sliced-Wasserstein distance [24].

5. Experiments

In general, a good distance metric is expected to have good performance in a wide range of downstream tasks. Here we compare the performance of different autoencoders trained with Chamfer discrepancy, Earth Mover's distance, sliced Wasserstein distance and max-sliced Wasserstein distance. We consider the following tasks in our evaluation: point cloud reconstruction, transfer learning, point cloud registration, and point cloud generation.

Implementation details. We follow the same architecture of the autoencoder used in [42], which is based on PointNet [44], with 256-dimensional embedding space. The architecture of our autoencoder is shown in Figure 2. We train

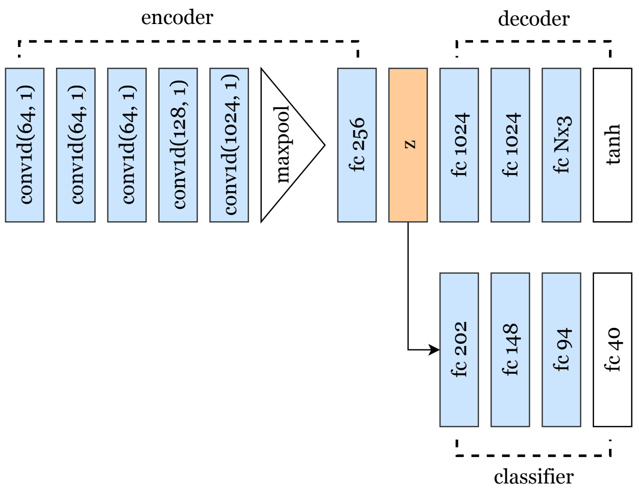

Figure 2: The network architecture of the autoencoder used in all of our experiments. The classifier is only used in the transfer learning experiment. All layers are followed by ReLU activation and batch normalization by default, except for the final layers.

Method	CD	SWD	EMD
CD-AE	0.014	6.738	0.314
EMD-AE	0.014	2.295	0.114
SSW-AE (ours)	**0.007**	**0.831**	**0.091**
MSW-AE (ours)	**0.007**	0.865	0.093
ASW-AE (ours)	**0.007**	0.854	0.092

Table 1: Quantitative measurements of the discrepancy between the input point clouds and their reconstructed versions on ModelNet40. We use Chamfer discrepancy (CD), sliced Wasserstein distance with 100 slices (SWD), and EMD as the evaluation metrics.

the autoencoder on the ShapeNet Core-55 dataset [6] for 300 epochs with the following loss functions: Chamfer discrepancy (CD-AE), Earth-mover distance (EMD-AE), max-sliced Wasserstein distance (MSW-AE) and sliced Wasserstein distance. In the case which the autoencoder trained with sliced Wasserstein distance, we conduct experiments for each of two algorithms: Monte Carlo estimation and ASW. We will call these two autoencoders SSW-AE and ASW-AE, respectively. For Monte Carlo estimation, we set the number of slices 100. For ASW, we set the parameters in Algorithm 1 as follows: $N_0 = 2$, $s = 1$, $\epsilon = 0.5$ and $M = 500$. Our models are trained with an SGD optimizer with an initial learning rate 0.001, a momentum of 0.9, and a weight decay of 0.0005. We use an NVIDIA V100 GPU for both training and evaluation, with batch size of 128 and a point cloud size of 2048.

Method	Accuracy (%)
CD-AE	83.9
EMD-AE	84.4
SSW-AE (ours)	**86.8**
MSW-AE (ours)	86.5
ASW-AE (ours)	**86.8**

Table 2: Classification performance of different autoencoders on ModelNet40 [50]. Our proposed SW models can learn a better latent representation compared to Chamfer and EMD.

3D point cloud reconstruction. We test the reconstruction capability of the autoencoders on the ModelNet40 dataset [50]. We measure the differences between the original point clouds and their reconstructed versions using Chamfer discrepancy (CD), sliced Wasserstein distance with 100 slices (SWD), and EMD. The results are shown in Table 1. Figure 3 shows the qualitative results from different autoencoders. As can be seen, CD-AE performs well when being evaluated by Chamfer discrepancy, but not by SWD and EMD. On the contrary, from Lemma 1, minimizing Wasserstein distance leads to minimizing Chamfer distance as well. Experiments with other Wasserstein metrics including generalized sliced Wasserstein [24] could be found in the supplementary material.

Transfer learning. We further evaluate the performance of the autoencoders by using their latent vectors as features for classification. Particularly, for an input 3D shape, we feed its point cloud into an autoencoder and extract the corresponding latent vector. This vector is then classified by a classifier trained on the de-facto 3D classification benchmark of ModelNet40 [50]. The architecture of the classifier is shown in Figure 2. The input is a 256-dimension feature vector and the output is a 40-dimension vector representing the prediction scores of 40 classes in ModelNet40. We train our networks for 500 epochs with a batch size of 256. We use an SGD optimizer with 0.001 learning rate, 0.9 momentum, and 0.005 weight decay. The classification results are shown in Table 2. As can be seen, autoencoders trained with sliced Wasserstein distance outperformed both Chamfer and EMD. We further investigate performance of classifiers in case point clouds in ModelNet40 are perturbed by noise. We find that features learned with SWD are the most robust to noise, outperforming both CD and EMD by about 3% of accuracy. Details can be found in the supplementary material.

Point cloud generation. Next, we evaluate our method on the point cloud generation task. Following [1], we split the chair category of ShapeNet into train/validation/test sets in a 85/5/10 ratio. We train the autoencoders using different distance metrics for 10^4 epochs. After that, we train a gener-

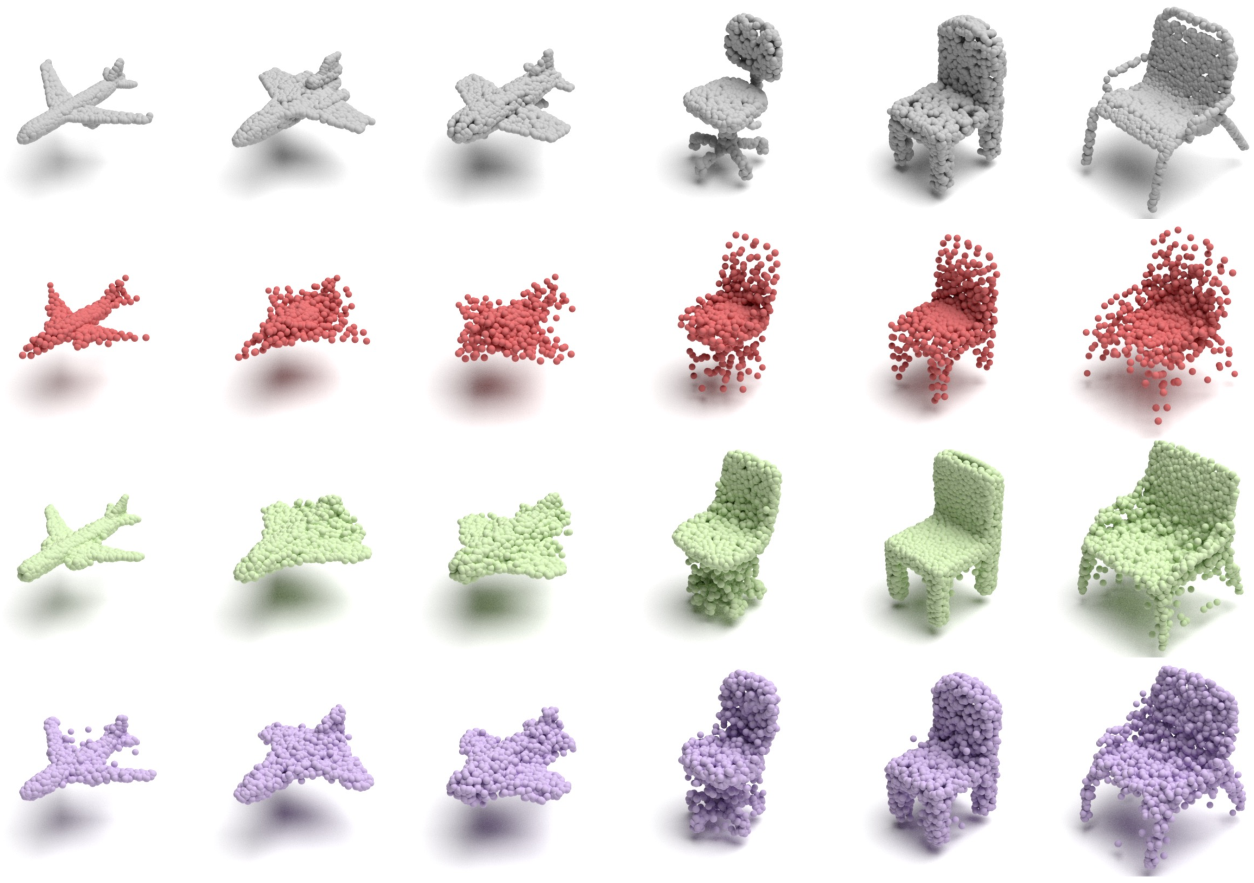

Figure 3: Qualitative results of autoencoders trained on single class using different loss functions. From top to bottom: input point clouds, CD-AE (red), EMD-AE (green) and SSW-AE (magenta).Compared to our models, CD-AE fails to reconstruct properly most of the 3D shapes.

Method	JSD (↓)	MMD (↓)		COV (%, ↑)		1-NNA (%, ↓)	
		CD	EMD	CD	EMD	CD	EMD
CD-AE	38.97	0.65	23.44	31.91	5.47	86.63	100.00
EMD-AE	3.73	**0.61**	**10.44**	**35.75**	35.75	**86.34**	**87.96**
SSW-AE (ours)	**3.24**	0.79	11.22	28.51	**37.96**	91.43	91.80

Table 3: Quantitative results of point cloud generation task on the chair category of ShapeNet. ↑: the higher the better, ↓: the lower the better. JSD, MMD-CD, and MMD-EMD scores are all multiplied by 10^2.

ator on the latent space of an autoencoder, same as [1]. Our generators, parameterized by a multi-layer perceptron, learn to map a 64-dimensional vector drawn from a normal distribution $\mathcal{N}(0, \mathbb{I}_{64})$ to a latent code learned by an autoencoder $\mathbb{P}_{\text{latent}}$, where $\mathbb{I}_{64}$ is the 64×64 identity matrix. We train the generators by minimizing the optimal transport distance between the generated and ground truth latent codes. We report the quantitative in Table 3.We use the same evaluation metrics as proposed by [51]. Qualitative results and results for MSW-AE and ASW-AE can be found in the supplementary.

3D point cloud registration. Finally, we consider the problem of 3D point cloud registration. In this problem, we need to estimate a rigid transformation between two 3D point clouds. We follow the same setup as [42] and use the autoencoders for local feature extraction. Evaluation is performed on the standard 3DMatch benchmark [54]. We also compare our models against CZK [7], a method that used geometric features. The final results are shown in Table 4. Our methods outperform other models by a good margin. Results for MSW-AE and ASW-AE can be found in the supplementary.

	CD-AE	EMD-AE	SSW-AE	CZK [7]
home1	59.4	**60.4**	**60.4**	63.2
home2	47.2	46.5	**47.8**	40.3
hotel1	62.6	62.1	**69.8**	64.3
hotel2	43.6	44.9	48.7	**66.7**
hotel3	46.2	34.6	**65.4**	57.7
kitchen	58.4	57.0	**62.6**	49.9
lab	42.2	46.7	**48.9**	37.8
study	50.4	50.0	**55.6**	54.7
Average	51.3	50.3	**57.4**	54.3

Table 4: 3D registration results (recall) on the 3DMatch benchmark. We compare the models that trained with sliced Wasserstein (SW-AE) and squared sliced Wasserstein (SSW-AE) against Chamfer discrepancy (CD-AE) and a geometric-based approach (CZK). More details can be found in the suppplementary.

Distance	Runtime (ms)
EMD	385
CD	120
SWD	138

Table 5: Training time per iteration in milliseconds of different distance functions. We compare the sliced Wasserstein distance (SWD) against Chamfer discrepancy (CD) and approximated EMD.

Runtime performance. We report the training time per iteration when training the autoencoder using Sliced Wasserstein with fixed number of slices, Chamfer and approximated EMD. We train over 10^4 iterations with a batch size of 128 and report the average timing. For Chamfer and EMD, we use the implementation from [51]. Otherwise, we use our Python implementation. Table 5 shows the runtime performance of different metrics. Our proposed sliced Wasserstein distance is as fast as Chamfer discrepancy, while being more accurate. Compared to EMD, sliced Wasserstein distance is almost three times faster.

Convergence rate. Our proposed sliced Wasserstein distance also has better convergence rate compared to Chamfer and EMD. To demonstrate this point, we calculate the error during training using exact EMD. Results in Table 6 show that all of the SW variants converge much faster than Chamfer and EMD.

Different architecture. We further performed experiments on point cloud capsule networks [55] to show that our method is agnostic to network architecture. As illustrated in Table 7, SWD is agnostic to network architecture we tested, i.e., SWD also works well for the point cloud capsule net-

Method	Epoch 50	Epoch 150	Epoch 300
CD-AE	0.268	0.275	0.314
EMD-AE	0.171	0.152	0.144
SSW-AE	**0.106**	**0.097**	**0.091**
MSW-AE	0.110	0.099	0.093
ASW-AE	0.109	0.098	0.092

Table 6: Convergence rate of different distance metrics during training. We report the exact EMD errors at epoch 50, 150 and 300. Sliced Wasserstein distance has the best convergence rate.

Method	CD	SWD	EMD	Accuracy
PCN-SSW	0.006	0.761	0.084	88.78
PCN-CD	0.003	3.035	0.156	88.45

Table 7: Quantitative measurements of the discrepancy between the input point clouds and their reconstructed versions on ModelNet40. The last column is the classification accuracy on ModelNet40.

work [55] (PCN), which has a very different design from the original PointNet. We trained PCN on ShapeNetCore55 dataset and tested on ModelNet40. As can be seen, using SWD can result in a slight performance improvement compared to Chamfer distance in the reconstruction task. Using SWD with PCN leads to a significant improvement of almost 2% in the classification task.

6. Conclusion

In the paper, we propose using sliced Wasserstein distance for learning representation of 3D point clouds. We theoretically demonstrate that the sliced Wasserstein distance is equivalent to EMD while its computational complexity is comparable to Chamfer divergence. Therefore, it possesses both the statistical and computational benefits of EMD and Chamfer divergence, respectively. We also propose a new algorithm to approximate sliced Wasserstein distance between two given point clouds so that the estimation is close enough to the true value. Empirically, we show that the latent codes of the autoencoders learned using sliced Wasserstein distance are more useful for various downstream tasks than those learned using the Chamfer divergence and EMD.

Acknowledgment. Tam Le acknowledges the support of JSPS KAKENHI Grant number 20K19873.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3D point clouds. In *ICML*, 2018. 1, 2, 3, 6, 7

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, pages 214–223, 2017. 2, 4

[3] Tristan Aumentado-Armstrong, Stavros Tsogkas, Allan Jepson, and Sven Dickinson. Geometric disentanglement for generative latent shape models. In *ICCV*, 2019. 3

[4] Erhan Bayraktar and Gaoyue Guo. Strong equivalence between metrics of Wasserstein type. *arXiv preprint arXiv:1912.08247*, 2019. 4

[5] Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and Radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 1(51):22–45, 2015. 1, 2, 4

[6] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015. 6

[7] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *CVPR*, 2015. 7, 8

[8] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *ICCV*, 2019. 1

[9] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, pages 2292–2300, 2013. 3

[10] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPF-FoldNet: Unsupervised learning of rotation invariant 3D local descriptors. In *ECCV*, 2018. 1, 2, 3

[11] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPFNet: Global context aware local features for robust 3D point matching. In *CVPR*, 2018. 1

[12] Haowen Deng, Tolga Birdal, and Slobodan Ilic. 3D local features for direct pairwise registration. In *CVPR*, pages 3244–3253, 2019. 1, 2, 3

[13] Ishan Deshpande, Yuan-Ting Hu, Ruoyu Sun, Ayis Pyrros, Nasir Siddiqui, Sanmi Koyejo, Zhizhen Zhao, David Forsyth, and Alexander G. Schwing. Max-sliced wasserstein distance and its use for gans. In *CVPR*, June 2019. 1, 2, 5

[14] Ishan Deshpande, Ziyu Zhang, and Alexander G. Schwing. Generative modeling using the sliced wasserstein distance. In *CVPR*, June 2018. 2, 4

[15] Chaojing Duan, Siheng Chen, and Jelena Kovacevic. 3D point cloud denoising via deep neural network based local surface estimation. In *ICASSP*, 2019. 1

[16] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3D object reconstruction from a single image. In *CVPR*, 2017. 1, 2, 3

[17] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3D point cloud matching with smoothed densities. In *CVPR*, 2019. 1

[18] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *CVPR*, 2018. 1

[19] Pedro Hermosilla, Tobias Ritschel, and Timo Ropinski. Total denoising: Unsupervised learning of 3D point cloud cleaning. In *ICCV*, 2019. 1, 2

[20] Nhat Ho, XuanLong Nguyen, Mikhail Yurochkin, Hung Hai Bui, Viet Huynh, and Dinh Phung. Multilevel clustering via Wasserstein means. In *ICML*, pages 1501–1509, 2017. 2

[21] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *CVPR*, 2018. 1

[22] Le Hui, Rui Xu, Jin Xie, Jianjun Qian, and Jian Yang. Progressive point cloud deconvolution generation network. In *ECCV*, 2020. 1

[23] Viet Huynh, Nhat Ho, Nhan Dam, Long Nguyen, Mikhail Yurochkin, Hung Bui, and Dinh Phung. On efficient multilevel clustering via Wasserstein distances. *Journal of Machine Learning Research*, pages 1–43, 2021. 2

[24] Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo Rohde. Generalized sliced wasserstein distances. In *NeurIPS*, 2019. 2, 5, 6, 11

[25] Soheil Kolouri, Phillip E. Pope, Charles E. Martin, and Gustavo K. Rohde. Sliced Wasserstein auto-encoders. In *ICLR*, 2019. 2, 4

[26] Soheil Kolouri, Gustavo K. Rohde, and Heiko Hoffmann. Sliced wasserstein distance for learning gaussian mixture models. In *CVPR*, June 2018. 4

[27] Soheil Kolouri, Yang Zou, and Gustavo K. Rohde. Sliced wasserstein kernels for probability distributions. In *CVPR*, June 2016. 4

[28] Rongjie Lai and Hongkai Zhao. Multi-scale non-rigid point cloud registration using robust sliced-wasserstein distance via laplace-beltrami eigenmap. *arXiv preprint arXiv: 1406.3758*, 2014. 4

[29] Trung Le, Tuan Nguyen, Nhat Ho, Hung Bui, and Dinh Phung. LAMDA: Label matching deep domain adaptation. In *ICML*, 2021. 2

[30] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *CVPR*, June 2019. 2

[31] Chun-Liang Li, Tomas Simon, Jason Saragih, Barnabas Poczos, and Yaser Sheikh. Lbs autoencoder: Self-supervised fitting of articulated meshes to point clouds. In *CVPR*, June 2019. 2

[32] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point cloud GAN. *arXiv preprint arXiv:1810.05795*, 2018. 1

[33] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3D object reconstruction. In *AAAI*, 2018. 1

[34] Tianyi Lin, Nhat Ho, and Michael Jordan. On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms. In *ICML*, pages 3982–3991, 2019. 3

[35] Tianyi Lin, Nhat Ho, and Michael I. Jordan. On the efficiency of Sinkhorn and Greenkhorn and their acceleration for optimal transport. *arXiv preprint arXiv: 1906.01437*, 2019. 3

[36] Kimia Nadjahi, Alain Durmus, Umut Simsekli, and Roland Badeau. Asymptotic guarantees for learning generative models with the sliced-wasserstein distance. In *NeurIPS*, pages 250–260, 2019. 4

[37] Khai Nguyen, Nhat Ho, Tung Pham, and Hung Bui. Distributional sliced-Wasserstein and applications to generative modeling. In *ICLR*, 2021. 2

[38] Khai Nguyen, Son Nguyen, Nhat Ho, Tung Pham, and Hung Bui. Improving relational regularized autoencoders with spherical sliced fused Gromov Wasserstein. In *ICLR*, 2021. 2

[39] O. Pele and M. Werman. Fast and robust earth mover's distance. In *ICCV*, 2009. 1, 3, 4

[40] Gabriel Peyré and Marco Cuturi. Computational optimal transport, 2020. 4

[41] Quang-Hieu Pham, Duc Thanh Nguyen, Binh-Son Hua, Gemma Roig, and Sai-Kit Yeung. JSIS3D: Joint semantic-instance segmentation of 3D point clouds with multi-task pointwise networks and multi-value conditional random fields. In *CVPR*, 2019. 1

[42] Quang-Hieu Pham, Mikaela Angelina Uy, Binh-Son Hua, Duc Thanh Nguyen, Gemma Roig, and Sai-Kit Yeung. LCD: Learned cross-domain descriptors for 2D-3D matching. In *AAAI*, 2020. 2, 5, 7

[43] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *CVPR*, 2019. 1

[44] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017. 1, 5

[45] Mark Rowland, Jiri Hron, Yunhao Tang, Krzysztof Choromanski, Tamás Sarlós, and Adrian Weller. Orthogonal estimation of wasserstein distances. In *AISTATS*, 2019. 4

[46] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3D point cloud generative adversarial network based on tree structured graph convolutions. In *ICCV*, 2019. 1

[47] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. PointGrow: Autoregressively learned point cloud generation with self-attention. In *WACV*, 2020. 1

[48] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. In *ICLR*, 2018. 2

[49] Jiqing Wu, Zhiwu Huang, Dinesh Acharya, Wen Li, Janine Thoma, Danda Pani Paudel, and Luc Van Gool. Sliced wasserstein generative models. In *CVPR*, 2019. 2, 4

[50] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, 2015. 6

[51] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. PointFlow: 3D point cloud generation with continuous normalizing flows. In *ICCV*, 2019. 1, 7, 8

[52] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. FoldingNet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, 2018. 1, 2, 3

[53] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-Net: Point cloud upsampling network. In *CVPR*, 2018. 3

[54] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3DMatch: Learning local geometric descriptors from RGB-D reconstructions. In *CVPR*, 2017. 7

[55] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 3D point capsule networks. In *CVPR*, 2019. 2, 8

Appendix

In this supplemental material, we further test the robustness of the proposed metrics (Section A), and then report numerical results on more metrics including adaptive-sliced Wasserstein (ASW), max sliced Wasserstein (MSW) and generalized sliced Wasserstein (GSW) (Section B). We also report additional results on the point cloud registration (Section C) and the point cloud generation task (Section D). Additionally, we also provide an evaluation of the number of slices used in the sliced Wasserstein (SSW) on the reconstruction, classification, and registration task.

A. Robustness

We conduct an experiment to compare robustness between Chamfer, EMD, and SWD. Particularly, we train autoencoder using Chamfer, EMD, and SWD respectively on ShapeNet, with point coordinates in $[-1, 1]$. At test time, we use ModelNet40, and the point clouds are perturbed by Gaussian noise $N(0, \sigma^2)$ with $\sigma \in [0.01, 0.05]$. We use the autoencoder to extract features from noisy point clouds and then input to learn a classifier. Figure 4 shows the performance of the classifier with increasing standard deviation values, where the experiments are carried out three times and then taken average. The solid lines demonstrate the case where we train the autoencoder with clean point clouds, while the dashed lines demonstrate the case where we train with noisy point clouds, i.e., we perturb ShapeNet in the same way as we do with ModelNet40. In both cases, Figure 4 shows that features learned with SWD are the most robust to noise, outperforming both CD and EMD by about 3% of accuracy. We also found that CD is less robust than EMD.

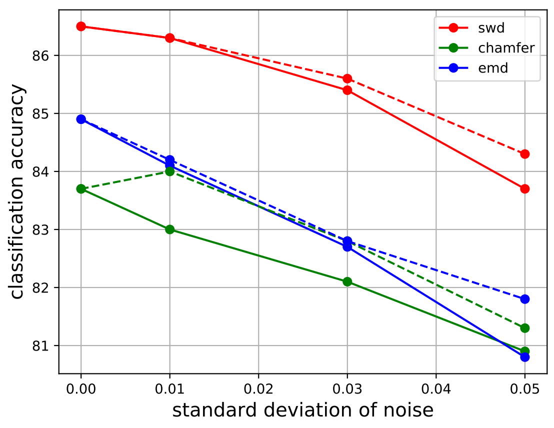

Figure 4: Classification accuracy on ModelNet40 with noisy data.

A.1. Performance with respect to batch size

We provide an experiment with batch size in Table 8, which shows negligible change in the Chamfer discrepancy between the input and reconstructed point clouds in ModelNet40 across batch sizes.

		Batch size		
		32	128	256
Model	SWD-AE	**0.006**	**0.007**	**0.008**
	CD-AE	0.012	0.014	0.014
	EMD-AE	0.012	0.014	0.013

Table 8: Average of the discrepancy between the input point clouds and their reconstructed versions on ModelNet40 with different batch sizes.

B. Generalized sliced Wasserstein distance

First, we recall briefly the definition of generalized sliced Wasserstein distance [24]. Generalized sliced-Wasserstein distance (GSW) extends the sliced-Wasserstein distance by replacing the inner product $\theta^\top x$ with a defining function $g(\theta, x)$ (cf. Assumptions H1-H4 in [24] for the definition of defining function). Denote $\pi_{g_\theta} \sharp \mu$ the pushforward measure of μ through the mapping g_θ where $g_\theta(x) := g(\theta, x)$ for all x. Then, for $p \geq 1$, the *GSW* is given by

$$GSW_p(\mu, \nu) := \left(\int_{\Omega_\theta} W_p^p(\pi_{g_\theta} \sharp \mu, \pi_{g_\theta} \sharp \nu) \right)^{1/p} \qquad (12)$$

where Ω_θ is the compact set of feasible parameters.
In our experiments, $\Omega_\theta := \mathbb{S}^2$ and $g(x, \theta) := ||x - \theta||_2$. To estimate GSW, we use Monte Carlo scheme as follows:

$$GSW_p(\mu, \nu) \approx \left(\frac{1}{N} \sum_{i=1}^{N} W_p^p(\pi_{g_{\theta_i}} \sharp \mu, \pi_{g_{\theta_i}} \sharp \nu) \right)^{\frac{1}{p}}. \qquad (13)$$

where we set $N := 100$. In Table 9, we provide numerical results for GSW on reconstruction and classification tasks. As we can see, GSW is slightly better than SW and MSW in reconstruction task, while SW is slightly better than other variants in classification task.

B.1. Effect of the number of slices

We measure the effect of varying the number of slices when computing sliced Wasserstein distance using Monte Carlo scheme. We denote SSW$n-$AE the auto-encoders trained using the sliced Wasserstein distance estimated by Monte Carlo estimation with n projections. We provide quantitative results for reconstruction and classification tasks in Table 10. Table 10 shows that increasing the number of slices in Monte Carlo estimation does not affect performance much in reconstruction and classification tasks.

Method	CD	SWD	EMD	Accuracy (%)
CD-AE	0.014	6.738	0.314	83.9
EMD-AE	0.014	2.295	0.114	84.4
SSW-AE (ours)	0.007	0.831	0.091	**86.8**
ASW-AE (ours)	0.007	0.854	0.092	**86.8**
MSW-AE (ours)	0.007	0.865	0.093	86.5
GSW-AE (ours)	**0.006**	**0.816**	**0.090**	85.8

Table 9: Quantitative measurements of the discrepancy between the input point clouds and their reconstructed versions on ModelNet40. The last column is the classification accuracy on ModelNet40.

Method	CD	SWD	EMD	Accuracy(%)
CD-AE	0.014	6.738	0.314	83.9
EMD-AE	0.014	2.295	0.114	84.4
SSW1-AE	**0.007**	0.901	0.094	86.5
SSW2-AE	**0.007**	0.865	0.093	86.5
SSW5-AE	**0.007**	0.829	**0.091**	86.7
SSW10-AE	**0.007**	**0.812**	**0.091**	**86.8**
SSW50-AE	**0.007**	0.849	0.092	**86.8**
SSW100-AE	**0.007**	0.831	**0.091**	**86.8**

Table 10: Quantitative measurements of the discrepancy between the input point clouds and their reconstructed versions on ModelNet40. The last column is the classification accuracy on ModelNet40.

C. Point cloud registration

In Table 11, we provide quantitative results as discussed in section Point cloud registration on page 7 of the main paper. Table 11 shows that SSW archives the best recall on average.

	ASW-AE	MSW-AE	GSW-AE	SSW-AE
home1	**63.2**	**63.2**	62.3	60.4
home2	49.1	49.1	**52.8**	47.8
hotel1	67.6	66.5	68.1	**69.8**
hotel2	46.2	**48.7**	46.2	**48.7**
hotel3	57.7	53.8	57.7	**65.4**
kitchen	**64.1**	63.3	63.3	62.6
lab	44.4	44.4	46.7	**48.9**
study	57.7	55.6	**58.5**	55.6
Average	56.3	55.6	57.0	**57.4**

Table 11: 3D registration results (recall) on the 3DMatch benchmark.

As in reconstruction, we measure the effect of varying the number of slices when computing sliced Wasserstein

	SSW1-AE	SSW5-AE	SSW10-AE	SSW50-AE	SSW100-AE	EMD-AE	CD-AE
home1	62.3	63.2	61.3	63.2	60.4	60.4	59.4
home2	49.7	48.4	49.1	50.9	47.8	46.5	47.2
hotel1	65.9	68.7	65.9	68.1	69.8	62.1	62.6
hotel2	50.0	43.6	43.6	47.4	48.7	44.9	43.6
hotel3	50.0	57.7	53.8	65.4	65.4	34.6	46.2
kitchen	64.4	62.6	63.7	62.1	62.6	57.0	58.4
lab	42.2	40.0	46.7	48.9	48.9	46.7	42.2
study	56.4	56.0	56.0	55.6	55.6	50.0	50.4
Average	55.1	55.0	55.0	**57.7**	<u>57.4</u>	50.3	51.3

Table 12: Varying number of slices for the 3D registration task. The best scores are highlighted in bold. The second best scores are underlined.

Method	JSD (↓)	MMD (↓)		COV (%, ↑)		1-NNA (%, ↓)	
		CD	EMD	CD	EMD	CD	EMD
SSW-AE	3.24	0.79	11.22	28.51	37.96	91.43	91.80
ASW-AE	3.58	0.73	10.65	31.76	36.48	93.57	93.94
MSW-AE	3.96	**0.59**	**9.64**	**35.89**	**40.47**	**89.73**	**89.29**
GSW-AE	**3.06**	0.76	10.98	30.13	37.52	91.21	91.65

Table 13: Quantitative results of point cloud generation task on the chair category of ShapeNet. ↑: the higher the better, ↓: the lower the better. JSD, MMD-CD, and MMD-EMD scores are all multiplied by 10^2.

distance for the registration task. The result is shown in Table 12. In the registration task, increasing the number of slices helps improve the performance by more than 2% on average (Table 12).

D. Point cloud generation

In Figure 5 and Table 13, we provide qualitative and quantitative results as mentioned in section Point cloud generation on page 6 of the main paper. As we can see, MSW archives best performance among SW variants in generation tasks.

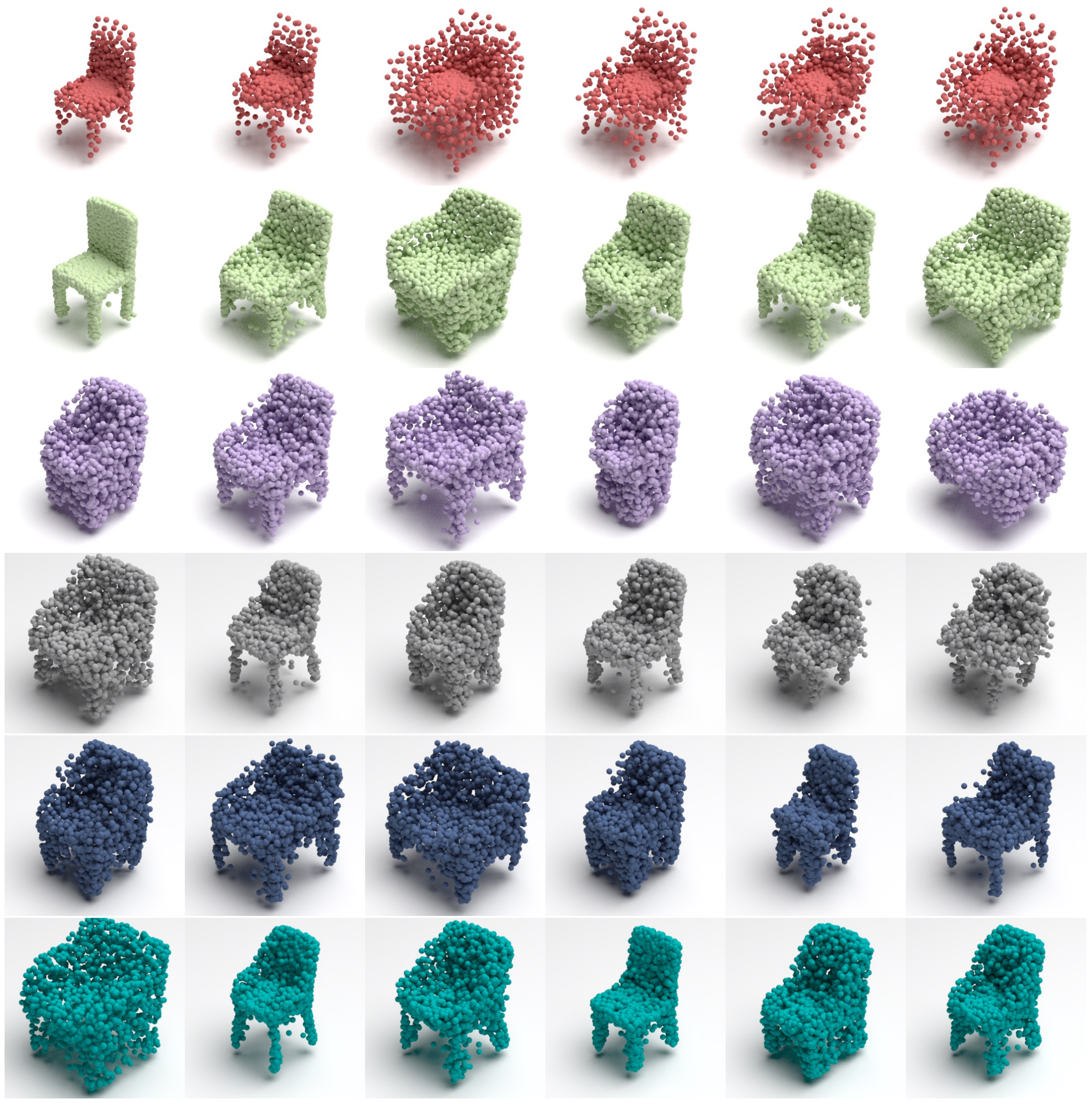

Figure 5: Point cloud generation results of the trained autoencoders on the chair category of ShapeNet. From top to bottom: CHAMFER-AE (red), EMD-AE (green) , SSW-AE (magenta), ASW-AE (gray) and MSW-AE (navy) and GSW-AE (aqua).