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]Singapore University of Technology and Design [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 a variant of the Wasserstein distance, named the sliced Wasserstein distance, for learning representations of 3D point clouds. Experiments show that the sliced Wasserstein distance allows the neural network to learn a more efficient representation compared to the Chamfer discrepancy. We demonstrate the efficiency of the sliced Wasserstein metric 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 [37, 20], object detection [36], and semantic segmentation [34]. Generative modeling with 3D point clouds has also been studied with some promising results [45, 39, 27, 21, 40, 28]. 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, 46, 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, as an effective metric 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 (cf. Equation (7)). 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 [33] 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.

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.
- An application of sliced Wasserstein distance as the reconstruction loss for unsupervised learning on 3D point clouds.
- An extensive evaluation of point cloud learning tasks including point cloud reconstruction, transfer learning,

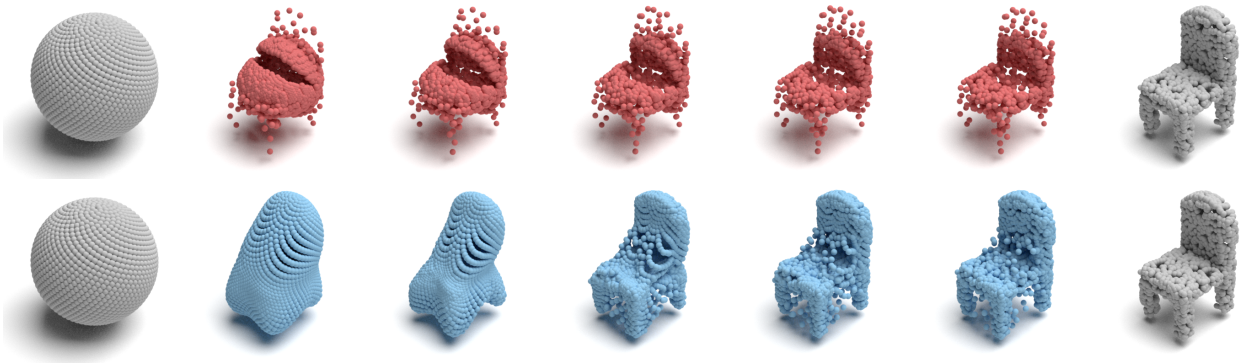


Figure 1: We advocate the use of sliced Wasserstein distance for training 3D point cloud autoencoders. In this example, we try to morph a sphere into a chair by optimizing two different loss functions: Chamfer discrepancy (top, red) and sliced Wasserstein distance (bottom, blue). The proposed sliced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

point cloud registration and generation based on Chamfer discrepancy, EMD and sliced Wasserstein distance.

2. Related Work

Set similarity. 3D point clouds autoencoders are useful in a wide range of applications, such as denoising [19], 3D matching [49, 10, 12], and generative models [16, 1]. Many autoencoder architectures have been proposed in recent few years [1, 46, 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, 46].

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 [35] 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, 41, 14, 13, 43]. In particular, Arjovsky et al. [2] proposed using Wasserstein as the loss function in generative adversarial networks (GANs) while Tolstikhin et al. [41] 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 linearly 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 [23]. 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 and improving the training stability over Wasserstein GAN. The follow-up work [13, 43] has an improved projection complexity compared to the sliced Wasserstein distance. In the recent work, Nguyen et al. [31] proposed a probabilistic approach to chooses a number of important directions via finding an optimal measure over the projections to greatly improve both the sliced and max-sliced Wasserstein distances. The authors also proposed another approach [32] to improve the sliced and max-sliced Wasserstein distances based on replacing the uniform distribution over slices by the von Mises Fisher distribution and then maximizing the location parameter of the von Mises Fisher distribution to identify the region of important directions. 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 [22]. 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 dis-

tance, 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 [46, 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, 47]. 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, 47] 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 $\max\{|P|, |Q|\}^3$ [33], 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 $\max\{|P|, |Q|\}^2$ [29], 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

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 strongly equivalent to Wasserstein distance and has efficient computation, as an alternative to EMD in Section 4.2.

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*

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

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

$$\begin{aligned} &\min_{y \in Q} \|x - y\|_2 \leq \|x - T(x)\|_2 \\ \Rightarrow &\min_{y \in Q} \|x - y\|_2^2 \leq K \|x - T(x)\|_2, \end{aligned}$$

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

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

Similarly we have $\sum_{y \in Q} \min_{x \in P} \|x - y\|_2^2 \leq K \sum_{y \in Q} \|y - \bar{T}(y)\|_2$ 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. 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 $\mathcal{O}(\max\{|P|^3, |Q|^3\})$ [33]. 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

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). 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 $p \geq 1$, the sliced Wasserstein distance of order p is formulated as follows:

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

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 strongly equivalent to first order WD or equivalently EMD [4], namely, there exist two positive constants C_1 and C_2 such that

$$C_1 \cdot SW_1(\mu, \nu) \leq EMD(\mu, \nu) \leq C_2 \cdot SW_1(\mu, \nu). \tag{7}$$

The strong 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 [26], generative models on 2D images; see, for examples [38, 30, 14, 23, 24, 43, 25]. 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 $\mathbb{S}^{\mathtt{d}-1}$ where $\mathtt{d}$ is the dimension of points in point clouds, which results in the following approximation of sliced Wasserstein distance:

$$SW_p(\mu, \nu) \approx \Big(\frac{1}{N} \sum_{i=1}^{N} W_p^p\big(\pi_{\theta_i} \sharp \mu, \pi_{\theta_i} \sharp \nu\big) \Big)^{\frac{1}{p}}. \tag{8}$$

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.

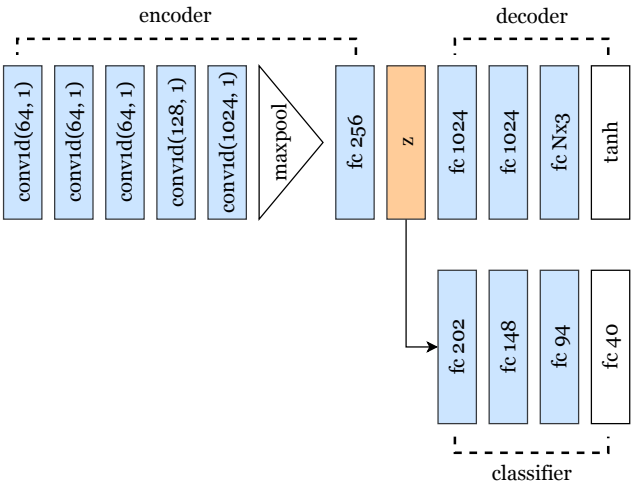


Figure 2: The network architecture of the autoencoder used in all of our experiments. The classifier is only used in the transfer learning experiment. All layers are followed by ReLU activation and batch normalization by default, except for the final layers.

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 (CD-AE), Earth Mover's distance (EMD-AE), sliced Wasserstein distance (SW-AE), and squared sliced Wasserstein distance (SSW-AE). We consider the following tasks in our evaluation: point cloud reconstruction, transfer learning, point cloud registration, and point cloud generation. Note that for training, we use an approximated version of EMD [16], while for testing, we calculate the exact EMD by linear programming.

Implementation details. We follow the same architecture of the autoencoder used in [35], which is based on PointNet [37], with 256-dimensional embedding space. The architecture of our autoencoder is shown in Figure 2. We train the autoencoder on the ShapeNet Core-55 dataset [6] for 300 epochs with the following loss functions: Chamfer discrepancy, Sliced Wasserstein distance, and squared Sliced Wasserstein distance. The two variants of Wasserstein distance use 100 slices in their computation. 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.

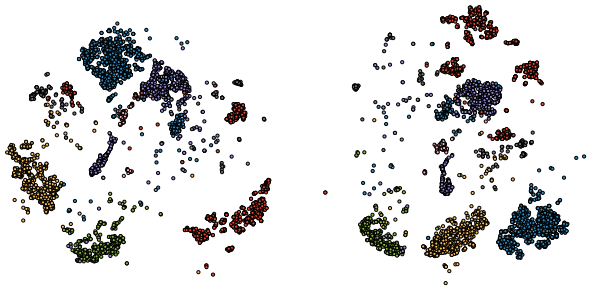
3D point cloud reconstruction. We test the reconstruction capability of the autoencoders on the ModelNet40 dataset [44]. We measure the differences between the original point clouds and their reconstructed versions using Chamfer discrepancy (CD), sliced Wasserstein distance with 100 slices (SWD), and EMD. The results are shown in Table 1. Figure 4 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. This is demonstrated by that SW-AE and SSW-AE perform well on all of the metrics. The squared variant SSW-AE slightly outperforms the vanilla SW-AE model on both Chamfer and EMD metrics.

Method	CD	SWD	EMD
CD-AE	0.014	6.738	0.264
EMD-AE	0.014	2.295	0.112
SW-AE (ours)	**0.007**	0.836	0.073
SSW-AE (ours)	**0.007**	**0.831**	**0.071**

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.

Method	Accuracy (%)
CD-AE	83.9
EMD-AE	84.4
SW-AE (ours)	86.3
SSW-AE (ours)	**86.8**

Table 2: Classification performance of different autoencoders on ModelNet40 [44]. Our proposed SW models can learn a better latent representation compared to Chamfer and EMD.



(a) CD-AE (b) SW-AE

Figure 3: t-SNE embeddings of CD-AE and SW-AE from the first 10 classes in the ModelNet40 dataset.

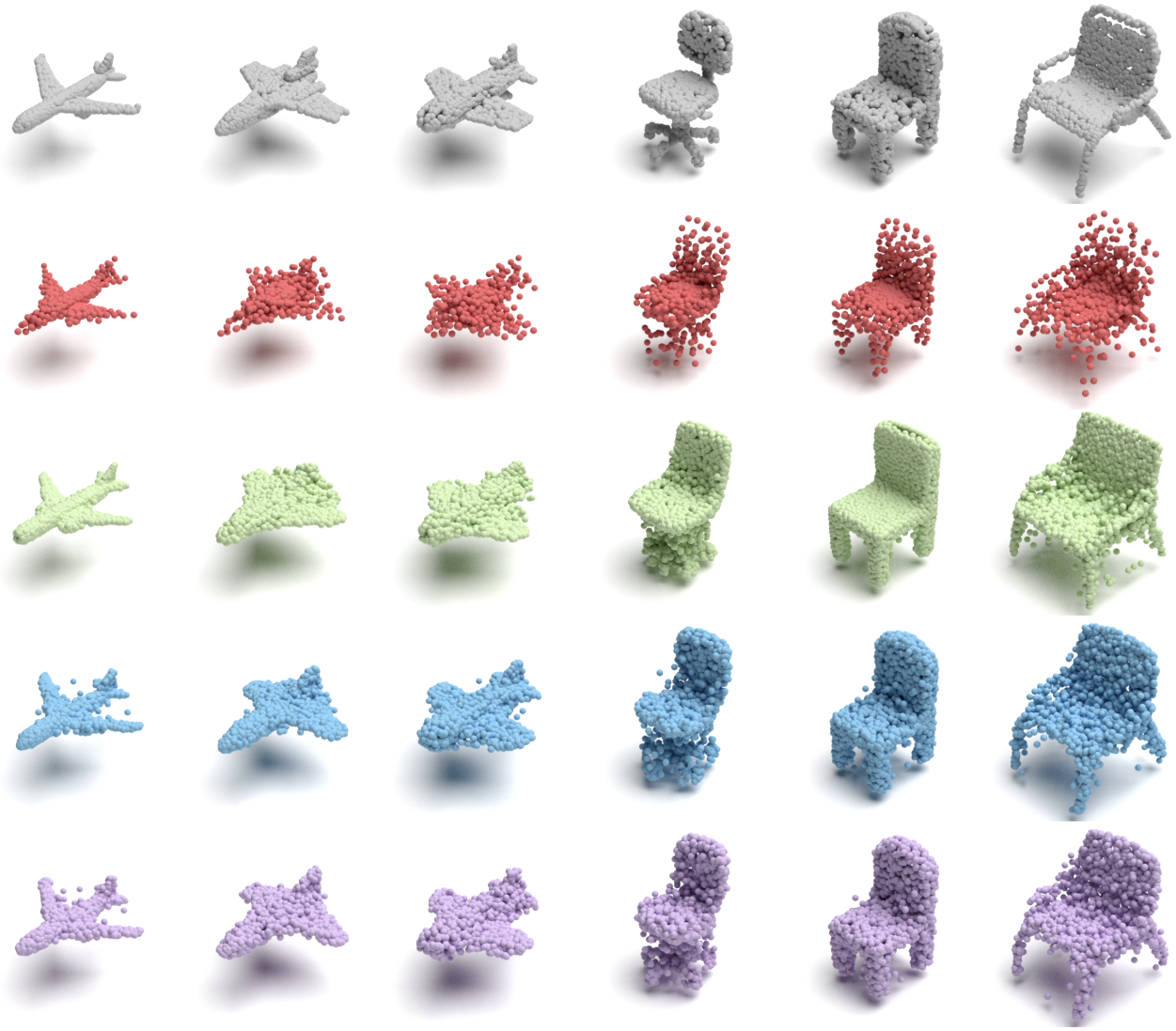


Figure 4: Qualitative results of autoencoders trained on single class using different loss functions. From top to bottom: input point clouds, CD-AE (red), EMD-AE (green), SW-AE100 (blue), and SSW-AE (magenta). Compared to our models, CD-AE fails to reconstruct properly most of the 3D shapes.

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 [44]. 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. In Figure 3, we visualize embeddings from trained autoencoders of the first 10 classes in ModelNet40 dataset by mapping the high-dimensional latent vector onto the 2D space using t-SNE [42].

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 generator on the latent space of an autoencoder, same as [1]. Our generators, parameterized by a multi-layer perceptron, learn

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**
SW-AE (ours)	3.82	0.78	11.15	27.33	33.97	92.32	92.39
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.

	CD-AE	SW-AE	SSW-AE	CZK [7]
home1	59.4	**65.1**	60.4	63.2
home2	47.2	**49.1**	47.8	40.3
hotel1	62.6	67.0	**69.8**	64.3
hotel2	43.6	48.7	48.7	**66.7**
hotel3	46.2	57.7	**65.4**	57.7
kitchen	58.4	61.0	**62.6**	49.9
lab	42.2	42.2	**48.9**	37.8
study	50.4	53.8	**55.6**	54.7
Average	51.3	55.6	**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).

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 and qualitative results in Figure 5 and Table 3, respectively. We use the same evaluation metrics as proposed by [45].

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 [35] and use the autoencoders for local feature extraction. Evaluation is performed on the standard 3DMatch benchmark [48]. 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.

Additional statistics

We now further provide additional results with runtime performance and convergence rate of sliced Wasserstein distance.

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.

Method	Epoch 50	Epoch 150	Epoch 300
CD-AE	0.220	0.227	0.264
EMD-AE	0.136	0.120	0.111
SW-AE	0.090	0.079	0.073
SSW-AE	**0.083**	**0.076**	**0.071**

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.

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 [45]. 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.

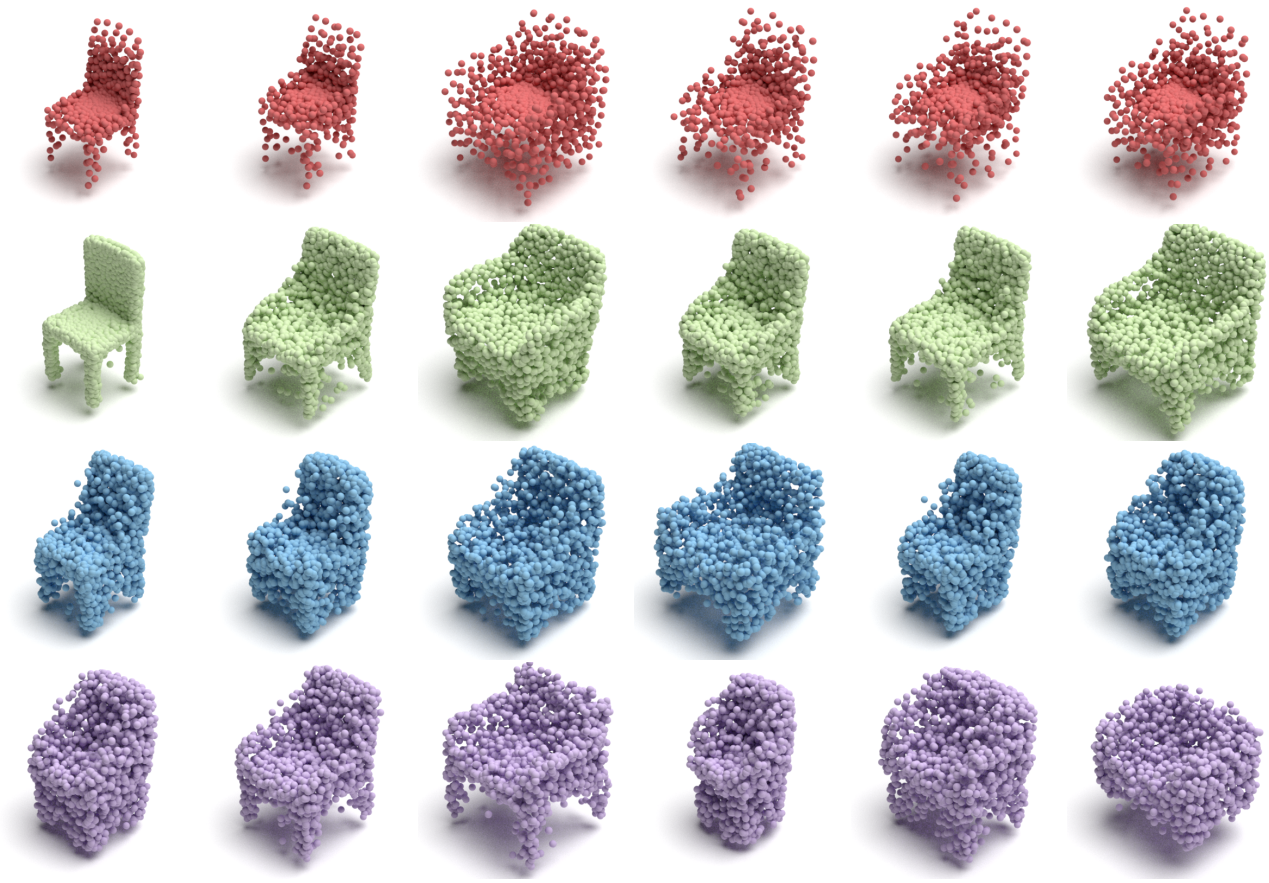


Figure 5: Point cloud generation results of the trained autoencoders on the chair category of ShapeNet. From top to bottom: CHAMFER-AE (red), EMD-AE (green), SW-AE (blue), and SSW-AE (magenta).

6. Conclusion

In the paper, we propose using sliced Wasserstein distance instead of Chamfer divergence and EMD 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. 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. We argue that the reason for this is that reconstruction quality of SW and its squared variant are higher than that of Chamfer divergence and EMD, so SW can help the autoencoder learns a more meaningful latent space.

Our work could potentially inspire deeper investigations on metrics for deep learning with 3D point clouds. For example, max-sliced Wasserstein [13] and generalized sliced Wasserstein [22] could be explored (see Appendix for our preliminary results). It is of great interest to study whether our findings can generalize to probabilistic autoencoders as well. From a practical point of view, simplifying the implementation of sliced Wasserstein and more efficient unit sphere sampling, *e.g.*, importance sampling and Gibbs sampling, could lead to wide adoption of this metric in downstream applications.

References

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

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017. 2

[3] Tristan Aumentado-Armstrong, Stavros Tsogkas, Allan Jepson, and Sven Dickinson. Geometric disentanglement for generative latent shape models. In *IEEE International Conference on Computer Vision*, 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. 5, 11

[7] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 7

[8] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *IEEE International Conference on Computer Vision*, 2019. 1

[9] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, pages 2292–2300, 2013. 3

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

[11] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPFNet: Global context aware local features for robust 3D point matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1

[12] Haowen Deng, Tolga Birdal, and Slobodan Ilic. 3D local features for direct pairwise registration. In *IEEE Conference on Computer Vision and Pattern Recognition*, 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 *IEEE Conference on Computer Vision and Pattern Recognition*, June 2019. 2, 8, 12

[14] Ishan Deshpande, Ziyu Zhang, and Alexander G. Schwing. Generative modeling using the sliced wasserstein distance. In *IEEE Conference on Computer Vision and Pattern Recognition*, 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 *IEEE International Conference on Acoustics, Speech and Signal Processing*, 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 *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 3, 5, 11

[17] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3D point cloud matching with smoothed densities. In *IEEE Conference on Computer Vision and Pattern Recognition*, 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 *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1

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

[20] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1

[21] Le Hui, Rui Xu, Jin Xie, Jianjun Qian, and Jian Yang. Progressive point cloud deconvolution generation network. In *European Conference on Computer Vision*, 2020. 1

[22] Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo Rohde. Generalized sliced wasserstein distances. In *NeurIPS*, 2019. 2, 8, 12

[23] Soheil Kolouri, Phillip E. Pope, Charles E. Martin, and Gustavo K. Rohde. Sliced Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2019. 2, 4

[24] Soheil Kolouri, Gustavo K. Rohde, and Heiko Hoffmann. Sliced wasserstein distance for learning gaussian mixture models. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018. 4

[25] Soheil Kolouri, Yang Zou, and Gustavo K. Rohde. Sliced wasserstein kernels for probability distributions. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016. 4

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

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

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

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

[30] Kimia Nadjahi, Alain Durmus, Umut Simsekli, and Roland Badeau. Asymptotic guarantees for learning generative models with the sliced-wasserstein distance. In *Advances in Neural Information Processing Systems*, pages 250–260, 2019. 4

[31] Khai Nguyen, Nhat Ho, Tung Pham, and Hung Bui. Distributional sliced-Wasserstein and applications to generative modeling. *arXiv preprint arXiv:2002.07367*, 2020. 2

[32] Khai Nguyen, Son Nguyen, Nhat Ho, Tung Pham, and Hung Bui. Improving relational regularized autoencoders with spherical sliced fused Gromov Wasserstein. *arXiv preprint arXiv:2010.01787*, 2020. 2

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

[34] 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 *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[35] 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 Conference on Artificial Intelligence*, 2020. 2, 5, 7

[36] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[37] 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*, 2017. 1, 5

[38] Mark Rowland, Jiri Hron, Yunhao Tang, Krzysztof Choromanski, Tamás Sarlós, and Adrian Weller. Orthogonal estimation of wasserstein distances. In *AISTATS*, 2019. 4, 12

[39] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3D point cloud generative adversarial network based on tree structured graph convolutions. In *IEEE International Conference on Computer Vision*, 2019. 1

[40] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. PointGrow: Autoregressively learned point cloud generation with self-attention. In *Winter Conference on Applications of Computer Vision*, 2020. 1

[41] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018. 2

[42] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. 6

[43] 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, 12

[44] 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*, 2015. 5, 6, 11, 12

[45] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. PointFlow: 3D point cloud generation with continuous normalizing flows. In *IEEE International Conference on Computer Vision*, 2019. 1, 7, 11

[46] 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*, 2018. 1, 2, 3

[47] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-Net: Point cloud upsampling network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 3

[48] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3DMatch: Learning local geometric descriptors from RGB-D reconstructions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 7

[49] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 3D point capsule networks. In *CVPR*, 2019. 2, 12

A. Appendix

A.1. Squared Sliced Wasserstein distance

We measure the effect of varying the number of projections when training the autoencoder using the Sliced Wasserstein (SW) and the squared Sliced Wasserstein (SSW) metric as discussed in the main paper. We show that even we only use 1 projection to evaluate the empirical sliced Wasserstein distance, the autoencoders trained with SW and SSW still outperform those trained with EMD or Chamfer.

Notion. We denote SWn − AE and SSWn − AE the autoencoders trained with the empirical Sliced Wasserstein distance and the empirical squared Sliced Wasserstein distance that use n projections, respectively. Also, we denote the autoencoders trained with Chamfer discrepancy and Earth Mover's distance CD − AE and EMD − AE, respectively.

Reconstruction. We train the autoencoder on the ShapeNet Core-55 dataset [6] for 300 epochs with the following loss functions: Chamfer discrepancy, Earth Mover's distance, Sliced Wasserstein distance, and squared Sliced Wasserstein distance. We test the reconstruction capability of the autoencoders on the ModelNet40 dataset [44]. 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 7. Note that for training, we use an approximated version of EMD [16], while for testing, we calculate the exact EMD by linear programming. The results for point cloud classification and point cloud registration are shown in Table 9 and Table 11, respectively.

Runtime performance. We report the training time per iteration when training the autoencoder using Sliced Wasserstein and squared Sliced Wasserstein, Chamfer and approximated EMD. We train over 10^3 iterations with a batch size of 128 and report the average timing. For Chamfer and EMD, we use the implementation from [45]. Otherwise, we use our Python implementation. We use an NVIDIA V100 GPU for testing. Table 8 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.

Summary. In our experiments, SSW slightly outperforms SW. Our hypothesis is that it is easier to optimize SSW than SW because taking the derivatives of a squared term is easier. More importantly, SW and SSW perform better than EMD.

Method	CD	SWD	EMD
CD-AE	0.014	6.738	0.264
EMD-AE	0.014	2.295	0.112
SW1-AE	0.0074	0.9401	0.0755
SW2-AE	0.0076	0.8522	0.0739
SW5-AE	0.0072	0.9149	0.0739
SW10-AE	0.0075	0.8596	0.0737
SW50-AE	0.0074	0.8538	0.0736
SW100-AE	0.0073	0.8360	0.0731
SSW1-AE	0.0069	0.9012	0.0727
SSW2-AE	<u>0.0068</u>	0.8645	0.0719
SSW5-AE	<u>0.0068</u>	<u>0.8280</u>	0.0711
SSW10-AE	0.0070	**0.8119**	0.0716
SSW50-AE	**0.0066**	0.8488	<u>0.0710</u>
SSW100-AE	<u>0.0068</u>	0.8312	**0.0707**

Table 7: 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 best scores are highlighted in bold. The second best scores are underlined.

Distance	Runtime (ms)
EMD	384
CD	<u>119</u>
SW1	<u>119</u>
SW2	<u>119</u>
SW5	<u>119</u>
SW10	120
SW50	127
SW100	138
SSW1	**118**
SSW2	**118**
SSW5	<u>119</u>
SSW10	<u>119</u>
SSW50	128
SSW100	138

Table 8: Training time per iteration in milliseconds of different distance functions. We compare the 2-sliced Wasserstein distance and the squared 2-sliced Wasserstein distance when varying the number of slices against Chamfer discrepancy (CD) and approximated EMD. The best scores are highlighted in bold. The second best scores are underlined.

From such empirical studies, we advocate the use of SW and SSW for point cloud learning.

For SW and SSW, the number of projection matters as

Method	Accuracy (%)
CD-AE	83.87 ± 0.41
EMD-AE	84.50 ± 0.31
SW1-AE	86.15 ± 0.20
SW2-AE	86.27 ± 0.23
SW5-AE	86.53 ± 0.24
SW10-AE	85.79 ± 0.19
SW50-AE	86.22 ± 0.35
SW100-AE	86.36 ± 0.27
SSW1-AE	86.60 ± 0.33
SSW2-AE	$\mathbf{86.79} \pm 0.27$
SSW5-AE	86.60 ± 0.23
SSW10-AE	$\underline{86.64} \pm 0.29$
SSW50-AE	86.50 ± 0.34
SSW100-AE	86.47 ± 0.21

Table 9: Classification performance of different autoencoders on ModelNet40 [44]. Our proposed SW models can learn a better latent representation compared to Chamfer and EMD. The best scores are highlighted in bold. The second best scores are underlined.

shown in the experiments. In our experiments, using SSW1-AE or SSW2-AE can provide very good performance in tasks such as classification (Table 9). However, using more projections like 50 or 100 is generally safe because using more projections requires only 20 milliseconds more in distance computation, which is negligible compared to entire training time (Table 8). More efficient ways to compute SW and SSW are left for future work.

A.2. Variants of Sliced Wasserstein distances

We have empirically explored various forms of Wasserstein distances (e.g., max-SWD [13], generalized SWD [22], orthogonal SWD [43, 38]). We observed that albeit simple, SWD is the most effective in our scenario. Finding more robust and effective Wasserstein distances for 3D point cloud deep learning is a promising future work where we believe SWD is a strong baseline. Details of the experiments are given below and in Table 10.

We use the scheme introduced in [22] to compute the max-sliced Wasserstein distance (MSSW) numerically. However, we found that max-sliced takes much too long to compute since it needs to solve the non-convex optimization problem to find the optimal projected direction. We compute Generalized SWD with circular defining function $g(x, \theta) = ||x-\theta||_2$ where $x \in \mathbb{R}^d, \theta \in \mathbb{S}^{d-1}$ [22], which we denote as GSSWn, where n is the number of slices. To approximate SW more effectively, we have tried to sample orthogonal projections instead of random projections [43, 38], which we denote as OSSWn. We also tried the variant of SW called Projected Wasserstein distance [38], which we denote as PWn. From the insights in the main paper, we use second order sliced Wasserstein distance and its variants with L2-norm ground metric (hence, SSW is for squared sliced Wasserstein). We used 1 random projection when computing all sliced Wasserstein distances for 3D point clouds since using more projections does not result in significant difference in performance. The only exception is the OSSW which requires 3 projections due to their orthogonalization process.

Method	CD	SWD	EMD	Accuracy
PointNet-SSW1	0.007	0.901	0.073	86.60
PointNet-CD	0.014	6.738	0.264	83.87
PointNet-GSSW1	0.007	0.861	0.072	86.26
PointNet-MSSW	0.007	0.865	0.072	86.27
PointNet-OSSW3	0.007	0.855	0.072	86.19
PointNet-PW1	0.102	8.616	0.335	82.09
PCN-SSW1	0.006	0.815	0.069	88.57
PCN-CD	0.003	3.146	0.129	88.41

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.

Autoencoder architectures. We further performed experiments on point cloud capsule networks to show that our method is agnostic to network architecture. As illustrated in Table 10, SWD is agnostic to network architecture we tested, i.e., SWD also works well for the point cloud capsule network [49] (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.

	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 11: 3D registration results (recall) on the 3DMatch benchmark. The best scores are highlighted in bold. The second best scores are underlined.

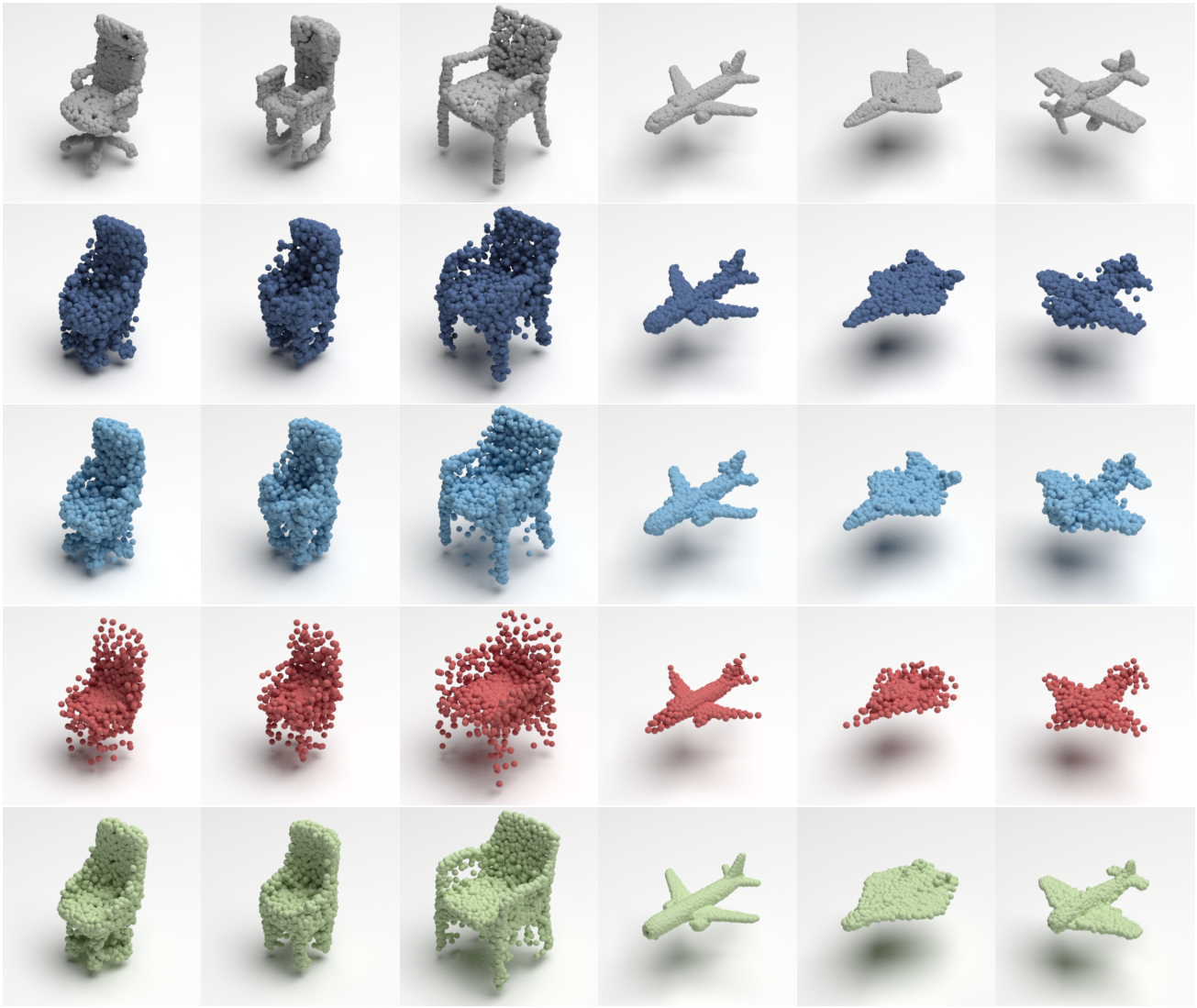


Figure 6: Qualitative results of autoencoders trained on single class over 10^4 epochs using SSW1, SSW100, Chamfer, EMD. From top to bottom: input point clouds, SSW1, SSW100, Chamfer, EMD.