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

Trung Nguyen\textsuperscript{1} Quang-Hieu Pham\textsuperscript{3} Tam Le\textsuperscript{4} Tung Pham\textsuperscript{1} Nhat Ho\textsuperscript{5} Binh-Son Hua\textsuperscript{1,2}
\textsuperscript{1}VinAI Research, Vietnam \textsuperscript{2}VinUniversity, Vietnam \textsuperscript{3}Woven Planet North America, Level 5 \textsuperscript{4}RIKEN AIP, Japan \textsuperscript{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 autoencoder, 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 $(\Omega, d)$ and two probability measures $\mu$ and $\nu$ on $\Omega$, let $\Pi(\mu, \nu)$ be the set of all joint distributions $\gamma$ such that its marginal distributions are $\mu$ and $\nu$, respectively. For any $\theta \in \mathbb{S}^{n-1}$ and any measure $\mu$, $\pi_{\theta} \nu$ denotes the pushforward measure of $\mu$ through the mapping $R_{\theta}$ where $R_{\theta}(x) = \theta^T x$ for all $x$.

3. Background

To study the performance of different metrics for point cloud learning, we briefly review the mathematical founda-
tion 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:

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

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:

\[
d_{\text{MCD}}(P, Q) = \max \left\{ \frac{1}{|P|} \sum_{x \in P} \min_{y \in Q} \|x - y\|_2, \frac{1}{|Q|} \sum_{y \in Q} \min_{x \in P} \|x - y\|_2 \right\}.
\]

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 \( (\Omega, d) \) be a metric space, \( \mu, \nu \) are probability measures on \( \Omega \). For \( p \geq 1 \), the \( p \)-Wasserstein distance (WD) is given by

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

For \( p = 1 \), the Wasserstein distance becomes the Earth Mover’s Distance (EMD), where the optimal joint distribution \( \gamma \) induces a map \( T : \mu \rightarrow \nu \), 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:

\[
W_p(\mu, \nu) = \left( \int_0^1 |F_X^{-1}(t) - F_Y^{-1}(t)|^p \, dt \right)^\frac{1}{p},
\]

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 \( \delta_x \) 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

\[
d_{\text{EMD}}(P, Q) = \min_{T : P \rightarrow Q} \sum_{x \in P} ||x - T(x)||_2.
\]

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
\[ d_{CD}(P, Q) \leq 2K d_{EMD}(P, Q). \]  

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

\[
\min_{y \in Q} \|x - y\|_2 \leq \|x - T(x)\|_2
\]

\[
\Rightarrow \min_{y \in Q} \|x - y\|_2^2 \leq K \|x - T(x)\|_2.
\]

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

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 \( \epsilon^2 \), 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 \( O(n \log n) \) where \( n \) is the maximum number of supports of the discrete probability measures \( \mu \) and \( \nu \). When \( p = 1 \), the \( SW_1 \) 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 (\( \leq 6 \)), sliced Wasserstein distance will still be able to retain useful information of the point clouds even after we project them to some direction. 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 \( S^{d-1} \) where \( d \) is the dimension of points in point clouds, which results in the following approximation.

**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 \( \mu \) and \( \nu \) on a direction, says \( \theta \), on the unit sphere to obtain two projected measures denoted by \( \pi_{\theta} \mu \) and \( \pi_{\theta} \nu \), respectively. Then, we compute the Wasserstein distance between two projected measures \( \pi_{\theta} \mu \) and \( \pi_{\theta} \nu \). 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 \( \theta \). In particular, for any given \( p \geq 1 \), the sliced Wasserstein distance of order \( p \) is formulated as follows:

\[
SW_p(\mu, \nu) = \left( \int_{\mathbb{S}^{d-1}} W_p^p(\pi_{\theta} \mu, \pi_{\theta} \nu) d\theta \right)^{\frac{1}{p}}. \tag{6}
\]

The \( SW_p \) is considered as a low-cost approximation for Wasserstein distance as its computational complexity is of the order \( O(n \log n) \) where \( n \) is the maximum number of supports of the discrete probability measures \( \mu \) and \( \nu \). When \( p = 1 \), the \( SW_1 \) 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 (\( \leq 6 \)), 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.
of sliced Wasserstein distance:
\[ SW_p(\mu, \nu) \approx \left( \frac{1}{N} \sum_{i=1}^{N} W_p^s(\pi_0, z_i \mu, \pi_0, z_i \nu) \right)^{\frac{1}{p}}. \] (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 \( \mu \) and \( \nu \). 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:
\[ M SW_p(\mu, \nu) := \max_{\theta \in \mathbb{R}^{n-1}} W_p(\pi_0, z_i \mu, \pi_0, z_i \nu). \] (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 \( \{\theta_i\}_{i}^{N} \) drawn from the sphere \( \mathbb{S}^{n-1} \), for the simplicity of presentation, we denote \( s \theta_i := W_p^s(\pi_0, z_i \mu, \pi_0, z_i \nu) \) for all \( 1 \leq i \leq N \). Furthermore, we denote \( \bar{s} \theta := SW_p^s(\mu, \nu) \) the true value of SW distance that we want to compute. The Monte Carlo estimation of the SW distance can be written as follows:
\[ \bar{s} \theta := \sum_{i=1}^{N} s \theta_i, \] which is an unbiased estimator of the true value \( \bar{s} \theta \). Similarly, the biased and unbiased variance estimates are respectively defined as:
\[ s_2^2 := \frac{1}{N} \sum_{i=1}^{N} (s \theta_i - \bar{s} \theta)^2, \quad \bar{s}_2^2 := \frac{N}{N-1} s_2^2. \] (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 \( \hat{N} \) such that the error bound is below a certain tolerance threshold. Particularly, applying central limit theorem, we have
\[ \mathbb{P}\left( |\bar{s} \theta - \bar{s} \theta| < \frac{k \bar{s}_2}{\sqrt{N}} \right) \approx \phi(k) - \phi(-k) \] (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 \( M \)

Output: \( \bar{s} \theta_N \)
Sample \( N_0 \) projections ;
Compute \( \bar{s} \theta := \bar{s} \theta_{N_0}, s \theta_i := s \theta_{N_0} \), \( N := N_0 \);
while \( s \theta^2 - (\bar{s} \theta)^2 > \frac{(N-1)\epsilon^2}{4} \) & \( N \leq M \) do
Sample \( s \) projections ;
Compute \( \bar{s} \theta_{s}, s \theta_i^2 \); Assign \( \bar{s} \theta := N \times \bar{s} \theta_{s} + \frac{s \theta_i}{N+s} \); Assign \( s \theta_i^2 := N \times s \theta_i^2 + \frac{s \theta_i}{N+s} \); Assign \( N := N + s \);
end

Given a predefined tolerance \( \epsilon > 0 \), we aim for
\[ \frac{k \bar{s}_2}{\sqrt{N}} \leq \epsilon, \quad \text{or} \quad \frac{k^2 s_2^2}{N} \leq \epsilon^2. \] (11)

From Equation (9), we note that \( \frac{s_2^2}{N} = \frac{s_2^2}{N-1} \) and so it is desirable to choose \( N \) such that \( k^2 s_2^2 / N \leq \epsilon^2 \). Rewriting the biased variance in Equation (9), we get \( s_2^2 = \frac{1}{N} \sum_{i=1}^{N} s \theta_i^2 - (\bar{s} \theta)^2 \). Denote \( s \theta_i^2 := \frac{1}{N} \sum_{i=1}^{N} s \theta_i^2 \), the condition becomes
\[ s \theta_i^2 - (\bar{s} \theta)^2 \leq \frac{(N-1)\epsilon^2}{k^2}. \]

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>CD</th>
<th>SWD</th>
<th>EMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD-AE</td>
<td>0.014</td>
<td>6.738</td>
<td>0.314</td>
</tr>
<tr>
<td>EMD-AE</td>
<td>0.014</td>
<td>2.295</td>
<td>0.114</td>
</tr>
<tr>
<td>SSW-AE (ours)</td>
<td>0.007</td>
<td>0.831</td>
<td>0.091</td>
</tr>
<tr>
<td>MSW-AE (ours)</td>
<td>0.007</td>
<td>0.865</td>
<td>0.093</td>
</tr>
<tr>
<td>ASW-AE (ours)</td>
<td>0.007</td>
<td>0.854</td>
<td>0.092</td>
</tr>
</tbody>
</table>

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.

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>JSD (↓)</th>
<th>MMD (↓)</th>
<th>COV (%, ↑)</th>
<th>1-NNA (%, ↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CD</td>
<td>EMD</td>
<td>CD</td>
<td>EMD</td>
</tr>
<tr>
<td>CD-AE</td>
<td>38.97</td>
<td>0.65</td>
<td>31.91</td>
<td>5.47</td>
</tr>
<tr>
<td>EMD-AE</td>
<td>3.73</td>
<td>0.61</td>
<td>35.75</td>
<td>35.75</td>
</tr>
<tr>
<td>SSW-AE (ours)</td>
<td>3.24</td>
<td>0.79</td>
<td>28.51</td>
<td>37.96</td>
</tr>
</tbody>
</table>

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 $\mathcal{N}(0, I_{64})$ to a latent code learned by an autoencoder $P_{\text{latent}}$, where $I_{64}$ is the $64 \times 64$ 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 3DMatching 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.
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.

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

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.

<table>
<thead>
<tr>
<th></th>
<th>EMD</th>
<th>CD</th>
<th>SWD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance</td>
<td>385</td>
<td>120</td>
<td>138</td>
</tr>
</tbody>
</table>

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 network.

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.

<table>
<thead>
<tr>
<th></th>
<th>Epoch 50</th>
<th>Epoch 150</th>
<th>Epoch 300</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD-AE</td>
<td>0.268</td>
<td>0.275</td>
<td>0.314</td>
</tr>
<tr>
<td>EMD-AE</td>
<td>0.171</td>
<td>0.152</td>
<td>0.144</td>
</tr>
<tr>
<td>SSW-AE</td>
<td>0.106</td>
<td>0.097</td>
<td>0.091</td>
</tr>
<tr>
<td>MSW-AE</td>
<td>0.110</td>
<td>0.099</td>
<td>0.093</td>
</tr>
<tr>
<td>ASW-AE</td>
<td>0.109</td>
<td>0.098</td>
<td>0.092</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th></th>
<th>CD</th>
<th>SWD</th>
<th>EMD</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCN-SSW</td>
<td>0.006</td>
<td>0.761</td>
<td>0.084</td>
<td>88.78</td>
</tr>
<tr>
<td>PCN-CD</td>
<td>0.003</td>
<td>3.035</td>
<td>0.156</td>
<td>88.45</td>
</tr>
</tbody>
</table>

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.

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.

References

[40] Gabriel Peyré and Marco Cuturi. Computational optimal transport, 2020. 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
[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] Yaqing Yang, Chen Feng, Yiru Shen, and Dong Tian. FoldingNet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, 2018. 1, 2, 3