Self-Attention Amortized Distributional Projection Optimization for Sliced Wasserstein Point-Cloud Reconstruction

Khai Nguyen [* 1]   **Dang Nguyen** [* 2]   **Nhat Ho** [1]

Abstract

Max sliced Wasserstein (Max-SW) distance has been widely known as a solution for less discriminative projections of sliced Wasserstein (SW) distance. In applications that have various independent pairs of probability measures, amortized projection optimization is utilized to predict the "max" projecting directions given two input measures instead of using projected gradient ascent multiple times. Despite being efficient, Max-SW and its amortized version cannot guarantee metricity property due to the sub-optimality of the projected gradient ascent and the amortization gap. Therefore, we propose to replace Max-SW with distributional sliced Wasserstein distance with von Mises-Fisher (vMF) projecting distribution (v-DSW). Since v-DSW is a metric with any non-degenerate vMF distribution, its amortized version can guarantee the metricity when performing amortization. Furthermore, current amortized models are not permutation invariant and symmetric. To address the issue, we design amortized models based on self-attention architecture. In particular, we adopt efficient self-attention architectures to make the computation linear in the number of supports. With the two improvements, we derive *self-attention amortized distributional projection optimization* and show its appealing performance in point-cloud reconstruction and its downstream applications.

1. Introduction

Wasserstein distance (Villani, 2008; Peyré & Cuturi, 2019) has been widely recognized in the community of machine learning as an effective tool. For example, Wasserstein distance is used to explore clusters inside data (Ho et al., 2017), to transfer knowledge between different domains (Courty et al., 2016; Damodaran et al., 2018), to learn generative models (Arjovsky et al., 2017; Tolstikhin et al., 2018), to extract features from graphs (Vincent-Cuaz et al., 2022), to compare datasets (Alvarez-Melis & Fusi, 2020), and many other applications. Despite being effective, Wasserstein distance is extremely expensive to compute. In particular, the computational complexity and memory complexity of Wasserstein distance in the discrete case is $\mathcal{O}(m^3 \log m)$ and $\mathcal{O}(m^2)$ respectively with m is the number of supports. The computational problem becomes more severe for applications that require computing the Wasserstein distance *multiple times* on different pairs of measures. Some examples can be named: deep generative modeling (Genevay et al., 2018), deep domain adaptation (Bhushan Damodaran et al., 2018), comparing datasets (Alvarez-Melis & Fusi, 2020), topic modeling (Huynh et al., 2020), point-cloud reconstruction (Achlioptas et al., 2018), and so on.

By adding entropic regularization (Cuturi, 2013), an ε-approximation of Wasserstein distance can be obtained in $\mathcal{O}(m^2/\varepsilon^2)$. However, this approach cannot reduce the memory complexity of $\mathcal{O}(m^2)$ due to the storage of the cost matrix. A more efficient approach is based on the closed-form solution of Wasserstein distance in one dimension which is known as sliced Wasserstein distance (Bonneel et al., 2015). Sliced Wasserstein (SW) distance is defined as the expectation of the Wasserstein distance between random one-dimensional push-forward measures from the two original measures. Thanks to the closed-form solution, SW can be solved in $\mathcal{O}(m \log_2 m)$ while having a linear memory complexity $\mathcal{O}(m)$. Moreover, SW is also better than Wasserstein distance in high-dimensional statistical inference. Namely, the sample complexity (statistical estimation rate) of SW is $\mathcal{O}(n^{-1/2})$ compared to $\mathcal{O}(n^{-1/d})$ of Wasserstein distance with d is the number dimension and n is the number of data samples. Due to appealing properties, SW is utilized successfully in various applications e.g., generative modeling (Deshpande et al., 2018; Nguyen & Ho, 2022b; Nguyen et al., 2023), domain adaptation (Lee et al., 2019a), Bayesian inference (Nadjahi et al., 2020; Yi & Liu, 2021), point-cloud representation learning (Nguyen et al., 2021c; Naderializadeh et al., 2021), and so on.

---

*Equal contribution [1]Department of Statistics and Data Sciences, University of Texas at Austin, USA [2]VinAI Research. Correspondence to: Khai Nguyen <khainb@utexas.edu>.

The downside of SW is that it treats all projections the same due to the usage of a uniform distribution over projecting directions. This choice is inappropriate in practice since there exist projecting directions that cannot discriminate two interested measures (Kolouri et al., 2018). As a solution, max sliced Wasserstein distance (Max-SW) (Deshpande et al., 2019) is introduced by searching for the best projecting direction that can maximize the projected Wasserstein distance. Max-SW needs to use a projected sub-gradient ascent algorithm to find the "max" slice. Therefore, in applications that need to evaluate Max-SW *multiple times* on *different pairs of measures*, the repeated optimization procedure is costly. For example, this paper focuses on point-cloud reconstruction applications where Max-SW needs to be computed between various pairs of empirical measures over a point-cloud and its reconstructed version.

To address the problem, amortized projection optimization is proposed in (Nguyen & Ho, 2022a). As in other amortized optimization (Shu, 2017; Amos, 2022) (learning to learn), an amortized model is estimated to predict the best projecting direction given the two input empirical measures. The authors in (Nguyen & Ho, 2022a) propose three types of amortized models including linear model, generalized linear model, and non-linear model. The linear model assumes that the "max" projecting direction is a linear combination of supports of two measures. The generalized linear model injects the linearity through a link function on the supports of two measures while the non-linear model uses multilayer perceptions to have more expressiveness.

Despite performing well in practice, the previous work has not explored the full potential of amortized optimization in the sliced Wasserstein setting. There are two issues in the current amortized optimization framework. Firstly, the sub-optimality of amortized optimization leads to losing the metricity of the projected distance from the predicted projecting direction. In particular, the metricity of Max-SW is only obtained at the global optimum. Therefore, using an amortized model with sub-optimal solutions cannot achieve the metricity for all pairs of measures. Losing metricity property could hurt the performance of downstream applications. Secondly, the current amortized models are not permutation invariant to the supports of two input measures and are not symmetric. The permutation-invariant and symmetry properties are vital since the "max" projecting direction is also not changed when permuting supports of two input empirical measures and exchanging two input empirical measures. By inducing the permutation-invariance and symmetry to the amortized model, it could help to learn a better amortized model and reduce the amortization gap

In this paper, we focus on overcoming the two issues of the current amortized projection optimization framework. For metricity preservation, we propose *amortized distribu-*

tional projection optimization framework which predicts the best distribution over projecting directions. In particular, we do amortized optimization for distributional sliced Wasserstein (DSW) distance (Nguyen et al., 2021a) with von Mises Fisher (vMF) slicing distribution (Jupp & Mardia, 1979) instead of Max-SW. Thanks to the smoothness of vMF, the metricity can be preserved even without a zero amortization gap. For the permutation-invariance and symmetry properties, we propose to use the self-attention mechanism (Vaswani et al., 2017) to design the amortized model. Moreover, we utilize efficient self-attention approaches that have the computational complexity scales linearly in the number of supports including efficient attention (Shen et al., 2021) and linear attention (Wang et al., 2020a).

Contribution. In summary, our contribution is two-fold:

1. First, we introduce *amortized distributional projection amortization* framework which predicts the best location parameter for von Mises-Fisher (vMF) distribution in distributional sliced Wasserstein (DSW) distance. Due to the smoothness of vMF, the metricity is guaranteed for all pairs of measures. Moreover, we enhance amortized models by inducing inductive biases which are permutation invariance and symmetry. To improve the efficiency, we leverage two linear-complexity attention mechanisms including efficient attention (Shen et al., 2021) and linear attention (Wang et al., 2020a) to parameterize the amortized model. Combining the above two improvements, we obtain *self-attention amortized distributional projection amortization* framework

2. Second, we adapt the new framework to the point-clouds reconstruction problem. In particular, we want to learn an autoencoder that can reconstruct (encode and decode) all point-clouds through their latent representations. The main idea is to treat a point-cloud as an empirical measure and use sliced Wasserstein distances as the reconstruction losses. Here, amortized optimization serves as a fast way to yield informative projecting directions for sliced Wasserstein distance to discriminative all pairs of original point-cloud and reconstructed point-cloud. Empirically, we show that the self-attention amortized distributional projection amortization provides better reconstructed point-clouds on the ModelNet40 dataset (Wu et al., 2015) than the amortized projection optimization framework and widely used distances. Moreover, on downstream tasks, the new framework also leads to higher classification accuracy on ModelNet40 and generates ShapeNet chairs with better quality.

Organization. The remainder of the paper is organized as follows. In Section 2, we provide backgrounds for point-cloud reconstruction and popular distances. In Section 3, we define the new amortized distributional projection optimization framework for the point-cloud reconstruction problem. Section 4 benchmarks the proposed method by extensive experiments on point-cloud reconstruction, transfer learning,
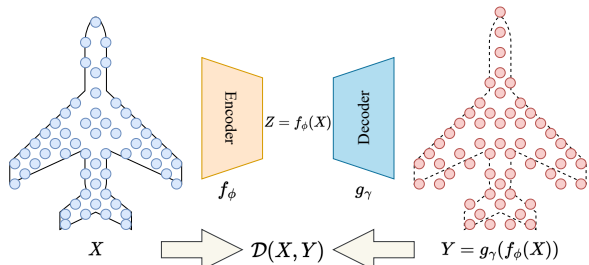
Figure 1. The reconstruction of a point-cloud X (a plane).

and point-cloud generation. Finally, proofs of key results and extra materials are in the supplementary.

Notation. For any $d \geq 2$, we denote $\mathcal{U}(\mathbb{S}^{d-1})$ is the uniform measure over the unit hyper-sphere $\mathbb{S}^{d-1} := \{\theta \in \mathbb{R}^d \mid ||\theta||_2^2 = 1\}$. For $p \geq 1$, $\mathcal{P}_p(\mathbb{R}^d)$ is the set of all probability measures on $\mathbb{R}^d$ that have finite p-moments. For any two sequences a_n and b_n, the notation $a_n = \mathcal{O}(b_n)$ means that $a_n \leq C b_n$ for all $n \geq 1$, where C is some universal constant. We denote $\theta \sharp \mu$ is the push-forward measures of μ through the function $f : \mathbb{R}^d \to \mathbb{R}$ that is $f(x) = \theta^\top x$.

2. Preliminaries

We first review the point-cloud reconstruction framework in Section 2.1. After that, we discuss famous choices of metrics between two point-clouds in Section 2.2. Finally, we present an adapted definition of the amortized projection optimization framework in the point-cloud reconstruction setting in Section 2.3.

2.1. Point-Cloud Reconstruction

We denote a point-cloud of m points $x_1, \ldots, x_m \in \mathbb{R}^d$ ($d \geq 1$) as $X = (x_1, \ldots, x_m) \in \mathbb{R}^{dm}$ which is a vector of a concatenation of all points in the point-cloud. We denote the set of all possible point-clouds as $\mathcal{X} \subset \mathbb{R}^{dm}$.

Permutation invariant metric space. Given a permutation one-to-one mapping function $\sigma : [m] \to [m]$, we have $\sigma(X) \in \mathcal{X}$ for all $X \in \mathcal{X}$. Moreover, we need a metric $\mathcal{D} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ such that $\mathcal{D}(X, \sigma(X)) = 0$ for all $X \in \mathcal{X}$ where $\sigma(X) = (x_{\sigma(1)}, \ldots, x_{\sigma(m)})$. Here, $\mathcal{D}$ is a metric, namely, it needs to satisfy the non-negativity, symmetry, triangle inequality, and identity property. The pair $(\mathcal{X}, \mathcal{D})$ forms a point-cloud metric space.

Learning representation via reconstruction. The raw representation of point-clouds is hard to work with in applications due to the complicated metric space. Therefore, a famous approach is to map point-clouds to points in a different space e.g., Euclidean, which is easier to apply machine learning algorithms. In more detail, we want to estimate a function $f_\phi : \mathcal{X} \to \mathcal{Z}$ ($\phi \in \Phi$) where $\mathcal{Z}$ is a set that belongs to another metric space. Then, we can apply machine learn-

ing algorithms on $\mathcal{Z}$ instead of $\mathcal{X}$. The most well-known and effective way to estimate the function f_ϕ is through reconstruction loss. Namely, we estimate f_ϕ jointly with a function $g_\gamma : \mathcal{Z} \to \mathcal{X}$ ($\gamma \in \Gamma$) given a point-cloud dataset $p(X)$ (distribution over $\mathcal{X}$) by minimizing the objective:

$$\min_{\phi \in \Phi, \gamma \in \Gamma} \mathbb{E}_{X \sim p(X)} \mathcal{D}(X, g_\gamma(f_\phi(X))). \quad (1)$$

The loss $\mathbb{E}_{X \sim p(X)} \mathcal{D}(X, g_\gamma(f_\phi(X)))$ is known as the reconstruction loss. If the reconstruction loss is 0, we have $g_\gamma = f_\phi^{-1}$ p-almost surely. Therefore, we can move from $\mathcal{X}$ to $\mathcal{Z}$ and move back from $\mathcal{Z}$ to $\mathcal{X}$ without losing information through the functions f_ϕ (referred as the encoder) and g_γ (referred as the decoder). We show an illustration of the framework (Achlioptas et al., 2018) in Figure 1. After learning how to do the reconstruction well, other point-cloud tasks can be done using the autoencoder (the pair (f_ϕ, g_γ)) e.g., shape interpolation, shape editing, shape analogy, shape completion, point-cloud classification, and point-cloud generation (Achlioptas et al., 2018).

2.2. Metric Spaces for Point-Clouds

We now review some famous choices of the metric $\mathcal{D}$ which are Chamfer distance (Barrow et al., 1977), Wasserstein distance (Villani, 2008), sliced Wasserstein (SW) distance (Bonneel et al., 2015), and max sliced Wasserstein (Max-SW) (Deshpande et al., 2019) distance.

Chamfer distance. For any two point-clouds X and Y, the Chamfer distance is defined as follows: $\text{CD}(X, Y) =$

$$\frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} ||x - y||_2^2 + \frac{1}{|Y|} \sum_{y \in Y} \min_{x \in X} ||x - y||_2^2, \quad (2)$$

where $|X|$ denotes the number of points in X.

Wasserstein distance. Given two probability measures $\mu \in \mathcal{P}_p(\mathbb{R}^d)$ and $\nu \in \mathcal{P}_p(\mathbb{R}^d)$, the Wasserstein distance between μ and ν is defined as follows:

$$W_p(\mu, \nu) = \left(\inf_{\pi \in \Pi(\mu, \nu)} \int_{\mathbb{R}^d \times \mathbb{R}^d} ||x - y||_p^p d\pi(x, y) \right)^{\frac{1}{p}} \quad (3)$$

where $\Pi(\mu, \nu)$ is set of all couplings whose marginals are μ and ν respectively. Since the Wasserstein distance is originally defined on probability measures space, we need to convert a point-cloud $X = (x_1, \ldots, x_m) \in \mathcal{X}$ to the corresponding empirical probability measure $P_X = \frac{1}{m} \sum_{i=1}^m \delta_{x_i} \in \mathcal{P}(\mathbb{R}^d)$. Therefore, we can use $\mathcal{D}(X, Y) = W_p(P_X, P_Y)$ for $X, Y \in \mathcal{X}$.

Sliced Wasserstein distance. As discussed, the Wasserstein distance is expensive to compute with the time complexity $\mathcal{O}(m^3 \log m)$ and the memory complexity $\mathcal{O}(m^2)$. Therefore, an alternative choice is sliced Wasserstein (SW) distance between two probability measures $\mu \in \mathcal{P}_p(\mathbb{R}^d)$ and

$\nu \in \mathcal{P}_p(\mathbb{R}^d)$ is:

$$\mathrm{SW}_p(\mu, \nu) = \left(\mathbb{E}_{\theta \sim \mathcal{U}(\mathbb{S}^{d-1})} \mathrm{W}_p^p(\theta \sharp \mu, \theta \sharp \nu)\right)^{\frac{1}{p}}, \quad (4)$$

The benefit of SW is that $\mathrm{W}_p(\theta \sharp \mu, \theta \sharp \nu)$ has a closed-form solution which is

$$\mathrm{W}_p(\theta \sharp \mu, \theta \sharp \nu) = \left(\int_0^1 |F_{\theta \sharp \mu}^{-1}(z) - F_{\theta \sharp \nu}^{-1}(z)|^p dz\right)^{\frac{1}{p}},$$

with F^{-1} denotes the inverse CDF function. The expectation is often approximated by Monte Carlo sampling, namely, it is replaced by the average from $\theta_1, \ldots, \theta_L$ that are drawn i.i.d from $\mathcal{U}(\mathbb{S}^{d-1})$. The computational complexity and memory complexity of SW becomes $\mathcal{O}(Lm \log_2 m)$ and $\mathcal{O}(Lm)$.

Max sliced Wasserstein distance. It is well-known that SW has a lot of less discriminative projections due to the uniform sampling. Therefore, max sliced Wasserstein distance is proposed to use the most discriminative projecting direction. Max sliced Wasserstein (Max-SW) distance (Deshpande et al., 2019) between $\mu \in \mathcal{P}_p(\mathbb{R}^d)$ and $\nu \in \mathcal{P}_p(\mathbb{R}^d)$ is introduced as follows:

$$\mathrm{Max\text{-}SW}_p(\mu, \nu) = \max_{\theta \in \mathbb{S}^{d-1}} W_p(\theta \sharp \mu, \theta \sharp \nu), \quad (5)$$

Max-SW is often computed by a projected sub-gradient ascent algorithm. When the projected sub-gradient ascent algorithm has $T \geq 1$ iterations, the computation complexity of Max-SW is $\mathcal{O}(Tm \log_2 m)$ and the memory complexity is $\mathcal{O}(m)$. Both SW and Max-SW are applied successfully in point-cloud reconstruction (Nguyen et al., 2021c).

2.3. Amortized Projection Optimization

Amortized Optimization. We start with the definition of amortized optimization.

Definition 1. For each context variable x in the context space $\mathcal{X}$, $\theta^\star(x)$ is the solution of the optimization problem $\theta^\star(x) = \arg\min_{\theta \in \Theta} \mathcal{L}(\theta, x)$, where Θ is the solution space. A parametric function $f_\psi : \mathcal{X} \to \Theta$, where $\psi \in \Psi$, is called an amortized model if

$$f_\psi(x) \approx \theta^\star(x), \quad \forall x \in \mathcal{X}. \quad (6)$$

The amortized model is trained by the amortized optimization objective which is defined as:

$$\min_{\psi \in \Psi} \mathbb{E}_{x \sim p(x)} \mathcal{L}(f_\psi(x), x), \quad (7)$$

where $p(x)$ is a probability measure on $\mathcal{X}$ which measures the "importance" of optimization problems.

Amortized Projection Optimization. We now revisit the point-cloud reconstruction objective with $\mathcal{D}(X, Y) =$

$\mathrm{Max\text{-}SW}_p(P_X, P_Y)$:

$$\min_{\phi \in \Phi, \gamma \in \Gamma} \mathbb{E}\left[\max_{\theta \in \mathbb{S}^{d-1}} \mathrm{W}_p(\theta \sharp P_X, \theta \sharp P_{g_\gamma(f_\phi(X))})\right], \quad (8)$$

where the expectation is with respect to $X \sim p(X)$. For each point-cloud $X \in \mathcal{X}$, we need to compute a Max-SW distance with an iterative optimization procedure. Therefore, it is computationally expensive.

Authors in (Nguyen & Ho, 2022a) propose to use amortized optimization (Shu, 2017; Amos, 2022) to speed up the problem. Instead of solving all optimization problems independently, an amortized model is trained to predict optimal solutions to all problems. In greater detail, given a parametric function $a_\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{S}^{d-1}$ ($\psi \in \Psi$), the amortized objective is:

$$\min_{\phi \in \Phi, \gamma \in \Gamma} \max_{\psi \in \Psi} \mathbb{E} \mathrm{W}_p(\theta_{\psi, \gamma, \phi} \sharp P_X, \theta_{\psi, \gamma, \phi} \sharp P_{g_\gamma(f_\phi(X))}), \quad (9)$$

where the expectation is with respect to $X \sim p(X)$, and $\theta_{\psi, \gamma, \phi} = a_\psi(X, g_\gamma(f_\phi(X)))$. The above optimization is solved by an alternative stochastic (projected)-gradient descent-ascent algorithm. Therefore, it is faster to compute in each update iteration of ϕ and γ. It is worth noting that the previous work (Nguyen & Ho, 2022a) considers the generative model application which is unstable and hard to understand. Here, we adapt the framework to the point-cloud reconstruction application which is easier to explore the behavior of amortized optimization. We refer the reader to Algorithms 2-3 in Appendix A.3 for algorithms on training an autoencoder with Max-SW and amortized projection optimization.

Amortized models. Authors in (Nguyen & Ho, 2022a) propose three types of amortized models that are based on the literature on linear models (Christensen, 2002). In particular, the linear amortized model is defined as:

Definition 2. Given $X, Y \in \mathbb{R}^{dm}$, the *linear amortized model* is defined as:

$$a_\psi(X, Y) := \frac{w_0 + X'w_1 + Y'w_2}{||w_0 + X'w_1 + Y'w_2||_2},$$

where X' and Y' are matrices of size $d \times m$ that are reshaped from the concatenated vectors X and Y of size dm, $\psi = (w_0, w_1, w_2)$ with $w_1, w_2 \in \mathbb{R}^m$, and $w_0 \in \mathbb{R}^d$.

Similarly, the generalized linear amortized model and the non-linear amortized model are defined by injecting non-linearity into the linear model. We review the definitions of the generalized linear amortized model and non-linear amortized model in Definitions 4-5 in Appendix A.1.

Sub-optimality. Despite being faster, amortized optimization often cannot recover the global optimum of optimization problems. Namely, we denote

$$\theta^\star(X) = \mathrm{argmax}_{\theta \in \mathbb{S}^{d-1}} \mathrm{W}_p(\theta \sharp P_X, \theta \sharp P_{g_\gamma(f_\phi(X))})$$
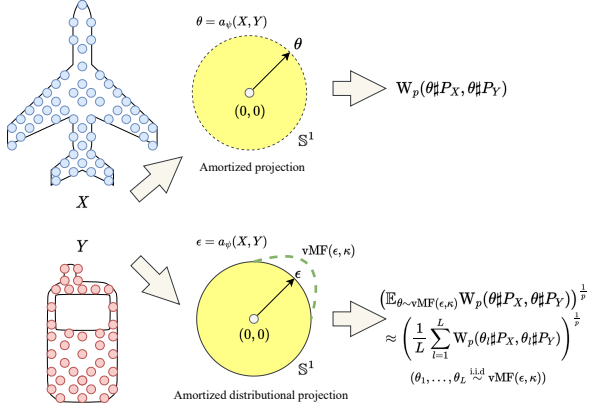
Figure 2. The difference between amortized projection optimization and amortized distributional projection optimization.

and $\psi^\star =$

$$\arg\max_{\psi \in \Psi} \mathbb{E}_{X \in p(X)} \left[\mathrm{W}_p(\theta_{\psi,\gamma,\phi} \sharp P_X, \theta_{\psi,\gamma,\phi} \sharp P_{g_\gamma(f_\phi(X))}) \right].$$

Then, it is well-known that the amortization gap $\mathbb{E}_{X \sim p(X)}[c(\theta^\star(X), a_{\psi^\star}(X, g_\gamma(f_\phi(X))))] > 0$ for a metric $c : \mathbb{S}^{d-1} \times \mathbb{S}^{d-1} \to \mathbb{R}^+$. A great amortized model is one that can minimize the amortization gap. However, in the amortized projection optimization setting, we cannot obtain $\theta^\star(X)$ since the projected gradient ascent algorithm can only yield the local optimum. Therefore, a careful investigation of the amortization gap is challenging.

3. Self-Attention Amortized Distributional Projection Optimization

In this section, we propose the self-attention amortized distributional projection optimization framework. First, we present amortized distributional projection optimization to maintain the metricity property in Section 3.1. We then introduce self-attention amortized models which are symmetric and permutation invariant in Section 3.2.

3.1. Amortized Distributional Projection Optimization

The current amortized projection optimization framework is for predicting the "max" projecting direction in Max-SW. However, the projected one-dimensional Wasserstein is only a metric on space of probability measure at the global optimum of Max-SW. Therefore, the local optimum from the projected sub-gradient ascent algorithm (Nietert et al., 2022) and the prediction from the amortized model only yield pseudo-metricity for the projected Wasserstein.

Proposition 1. *Let the projected one-dimensional Wasserstein be $PW_p(\mu, \nu; \hat{\theta}) = W_p(\hat{\theta}\sharp\mu, \hat{\theta}\sharp\nu))$ for any $\mu, \nu \in \mathcal{P}_p(\mathbb{R}^d)$ ($p \geq 1, d \geq 1$) and $\hat{\theta} \in \mathbb{S}^{d-1}$ such that $\hat{\theta} \neq \arg\max_{\theta \in \mathbb{S}^{d-1}} W_p(\theta\sharp\mu, \theta\sharp\nu)$, $PW_p(\mu, \nu; \hat{\theta})$ is a pseudo*

metric on $\mathcal{P}_p(\mathbb{R}^d)$ since it satisfies symmetry, non-negativity, triangle inequality, $\mu = \nu$ implies $PW_p(\mu, \nu; \hat{\theta}) = 0$, however, $PW_p(\mu, \nu; \hat{\theta}) = 0$ does not imply $\mu = \nu$.

The proof for Proposition 1 is given in Appendix B.1. This result implies that the if reconstruction loss $\mathbb{E}_{X \sim p(X)}[PW_p(P_X, P_{g_\gamma(f_\phi(X))}; \hat{\theta}(X)) = 0$, it does not imply $X = g_\gamma(f_\phi(X))$ for p-almost surely $X \in \mathcal{X}$. Therefore, a local maximum for $\max_{\theta \in \mathbb{S}^{d-1}}$ in Max-SW reconstruction (Equation 8) and the global maximum for $\max_{\psi \in \Psi}$ in amortized Max-SW reconstruction (Equation 9 with a misspecified amortized model) cannot guarantee perfect reconstruction even when their objectives obtain 0 values.

Amortized Distributional Projection Optimization. To overcome the issue, we propose to replace Max-SW in Equation 8 with the von Mises Fisher distributional sliced Wasserstein (v-DSW) distance (Nguyen et al., 2021b):

$$\min_{\phi \in \Phi, \gamma \in \Gamma} \mathbb{E}_{X \sim p(X)} \left[\max_{\epsilon \in \mathbb{S}^{d-1}} \left(\mathbb{E}_{\theta \sim \mathrm{vMF}(\epsilon,\kappa)} \right. \right.$$
$$\left. \left. \mathrm{W}_p^p(\theta\sharp P_X, \theta\sharp P_{g_\gamma(f_\phi(X))}) \right)^{\frac{1}{p}} \right], \quad (10)$$

where $\mathrm{vMF}(\epsilon, \kappa)$ is the von Mises Fisher distribution with the mean location parameter $\epsilon \in \mathbb{S}^{d-1}$ and the concentration parameter $\kappa > 0$, and v-DSW$_p(\mu, \nu; \kappa) = \max_{\epsilon \in \mathbb{S}^{d-1}} \left(\mathbb{E}_{\theta \sim \mathrm{vMF}(\epsilon,\kappa)} \mathrm{W}_p^p(\theta\sharp\mu, \theta\sharp\nu) \right)^{\frac{1}{p}}$ is von Mises Fisher distributional sliced Wasserstein distance. The optimization can be solved by a stochastic projected gradient ascent algorithm with the vMF reparameterization trick. In particular, $\theta_1, \ldots, \theta_L$ ($L \geq 1$) is sampled i.i.d from $\mathrm{vMF}(\epsilon, \kappa)$ via the reparameterized acceptance-rejection sampling (Davidson et al., 2018a) to approximate $\nabla_\epsilon \mathbb{E}_{\mathrm{vMF}(\epsilon,\kappa)}[\mathrm{W}_p^p(\theta\sharp\mu, \theta\sharp\nu)]$ via Monte Carlo integration. We refer the reader to Section A.2 for more detail about the vMF distribution, its sampling algorithm, its reparameterization trick, and the stochastic gradient estimators. We present a visualization of the difference between the new amortized distributional projection optimization framework and the conventional amortized projection optimization framework in Figure 2. The corresponding amortized objective is:

$$\min_{\phi \in \Phi, \gamma \in \Gamma} \max_{\psi \in \Psi} \mathbb{E}_{X \sim p(X)} \left(\mathbb{E}_{\theta \sim \mathrm{vMF}(\epsilon_{\psi,\gamma,\phi},\kappa)} \right.$$
$$\left. \mathrm{W}_p^p(\theta\sharp P_X, \theta\sharp P_{g_\gamma(f_\phi(X))}) \right)^{\frac{1}{p}}, \quad (11)$$

where $\epsilon_{\psi,\gamma,\phi} = a_\psi(X, g_\gamma(f_\phi(X)))$. The optimization is solved by an alternative stochastic (projected)-gradient descent-ascent algorithm with the vMF reparameterization.

Theorem 1. *For any $\epsilon \in \mathbb{S}^{d-1}$ and $0 \leq \kappa < \infty$, if $\mathbb{E}_{X \sim p(X)} \left(\mathbb{E}_{\theta \sim vMF(\epsilon,\kappa)} W_p^p(\theta\sharp P_X, \theta\sharp P_{g_\gamma(f_\phi(X))}) \right)^{\frac{1}{p}} = 0$, $X = g_\gamma(f_\phi(X))$ for p-almost surely $X \in \mathcal{X}$.*
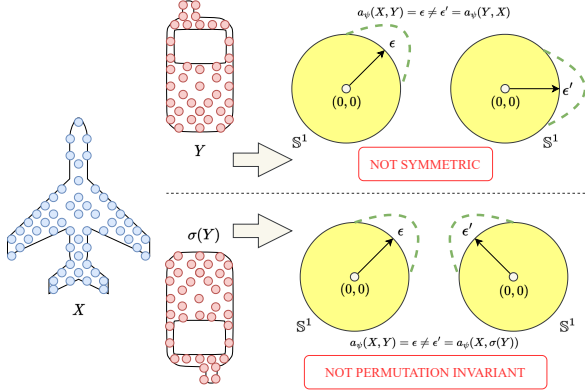
Figure 3. Visualization of an amortized model that is not symmetric and permutation invariant in two dimensions.

The proof of Theorem 1 is given in Appendix B.2. The proof is based on proving the metricity of the *non-optimal* von Mises Fisher distributional sliced Wasserstein distance (v-DSW) with the smoothness condition of the vMF distribution. It is worth noting that the proof of metricity of von Mises Fisher distributional sliced Wasserstein distance is new since the original work (Nguyen et al., 2021b) only shows the pseudo-metricity with the global optimality condition. Theorem 1 indicates that a perfect reconstruction can be obtained with a local optimum for $\max_{\epsilon \in \mathbb{S}^{d-1}}$ in v-DSW reconstruction (Equation 10) and a local optimum for $\max_{\psi \in \Psi}$ in amortized v-DSW reconstruction (Equation 11).

Comparison with SW and Max-SW: When $\kappa \to 0$, the vMF distribution converges weakly to the uniform distribution over the unit hypersphere. Hence, we can get back the conventional sliced Wasserstein reconstruction in both Equation 10 and Equation 11. When $\kappa \to \infty$, vMF distribution converges weakly to the Dirac delta at the location parameter. Therefore, we obtain Max-SW reconstruction and amortized Max-SW reconstruction in Equation 10 and Equation 11, respectively. However, when $0 < \kappa < \infty$, v-DSW reconstruction and amortized v-DSW reconstruction can find a region of discriminative projecting directions while preserving the metricity for perfect reconstruction.

3.2. Self-Attention Amortized Models

We now discuss the parameterization of the amortized model for amortized optimization.

Permutation Invariance and Symmetry. Let X and Y be two point-clouds, the optimal slicing distribution vMF$(\epsilon^\star, \kappa)$ of v-DSW between P_X and P_Y can be obtained by running Algorithm 4 in Appendix A.3. Clearly, vMF$(\epsilon^\star, \kappa)$ is invariant to the permutation of the supports since $P_{\sigma(X)} = P_X$ and $P_{\sigma(Y)} = P_Y$ for a permutation function σ. Moreover, the optimal slicing distribution vMF$(\epsilon^\star, \kappa)$ is also unchanged when we exchange P_X and P_Y since v-DSW is symmetric. However, the cur-

rent amortized models (see Definition 2, Definitions 4-5 in Appendix A.1) are not permutation invariant and symmetric, namely, $a_\psi(X, Y) \neq a_\psi(X, \sigma(Y))$ and $a_\psi(X, Y) \neq a_\psi(Y, X)$. Therefore, the current amortized models could be strongly misspecified. We show a visualization of an amortized model that is not symmetric and permutation invariant in Figure 3. To address the issue, we propose amortized models that are symmetric and permutation invariant based on the self-attention mechanism.

Self-Attention Mechanism. Attention is well-known for its effectiveness in learning long-range dependencies when data are sequences such as text (Devlin et al., 2019; Liu et al., 2019; Brown et al., 2020) or speech (Li et al., 2019; Wang et al., 2020b). This mechanism was then successfully generalized to other data types including image (Carion et al., 2020; Dosovitskiy et al., 2020), video (Sun et al., 2019), graph (Dwivedi & Bresson, 2021), point-cloud (Zhao et al., 2021; Guo et al., 2021), to name a few. We now revisit the attention mechanism (Vaswani et al., 2017). Given $Q, K \in \mathbb{R}^{m \times d_k}, V \in \mathbb{R}^{m \times d_v}$, the *scaled dot-product attention* operator is defined as:

$$\text{Att}(Q, K, V) = \text{softmax}_{\text{row}} \left[\frac{QK^T}{\sqrt{d_k}} \right] V \qquad (12)$$

where softmax$_{\text{row}}$ denotes the row-wise softmax function. In the self-attention mechanism, the query matrix Q, the key matrix K, and the value matrix V are usually computed by projecting the input sequence X into different subspaces. Thus, the self-attention mechanism is given as follows. Given $X \in \mathbb{R}^{m \times d}$, the *self-attention* operator is:

$$\mathcal{A}_\zeta(X) = \text{Att}(XW_q, XW_k, XW_v) \qquad (13)$$

where $W_q, W_k \in \mathbb{R}^{d \times d_k}, W_v \in \mathbb{R}^{d \times d_v}$ and $\zeta = (W_q, W_k, W_v)$. The self-attention operator is infamous for its quadratic memory and computational costs. In particular, given an input sequence of length m, both the time and space complexity are $\mathcal{O}(m^2)$. Since we focus on the sliced Wasserstein setting where the computational complexity should be at most $\mathcal{O}(m \log m)$, the conventional self-attention is not appropriate. Several works (Li et al., 2020; Katharopoulos et al., 2020; Wang et al., 2020a; Shen et al., 2021) have been proposed to reduce the overall complexity from $\mathcal{O}(m^2)$ to $\mathcal{O}(m)$. In this paper, we utilize two linear complexity variants of attention which are efficient attention (Shen et al., 2021) and linear attention (Wang et al., 2020a). Given $X \in \mathbb{R}^{m \times d}$, the *efficient self-attention* is defined as:

$$\mathcal{E}\mathcal{A}_\zeta(X) =$$
$$\text{softmax}_{\text{row}}(XW_q) \left[\text{softmax}_{\text{col}}(XW_k)^T (XW_v) \right] \qquad (14)$$

where $W_q, W_k \in \mathbb{R}^{d \times d_k}, W_v \in \mathbb{R}^{d \times d_v}, \zeta = (W_q, W_k, W_v)$, and softmax$_{\text{col}}$ denotes applying the soft-

max function column-wise. The *linear self-attention* is:

$$\mathcal{LA}_\zeta(X) = \text{Att}(XW_q, W_{k1}XW_{k2}, W_{v1}XW_{v2}) \quad (15)$$

where $W_q, W_{k2} \in \mathbb{R}^{d \times d_k}, W_{v2} \in \mathbb{R}^{d \times d_v}, W_{k1}, W_{v1} \in \mathbb{R}^{k \times n}$, and $\zeta = (W_q, W_{k1}, W_{k2}, W_{v1}, W_{v2})$. The projected dimension k is chosen such that $m \gg k$ to reduce the memory and space consumption significantly.

Self-Attention Amortized Models: Based on the self-attention mechanism, we introduce the self-attention amortized model which is permutation invariant and symmetric. Formally, the *self-attention amortized model* is defined as:

Definition 3. Given $X, Y \in \mathbb{R}^{dm}$, the *self-attention amortized model* is defined as:

$$a_\psi(X, Y) = \frac{\mathcal{A}_\zeta(X'^\top)^\top \mathbf{1}_m + \mathcal{A}_\zeta(Y'^\top)^\top \mathbf{1}_m}{||\mathcal{A}_\zeta(X'^\top)^\top \mathbf{1}_m + \mathcal{A}_\zeta(Y'^\top)^\top \mathbf{1}_m||_2}, \quad (16)$$

where X' and Y' are matrices of size $d \times m$ that are reshaped from the concatenated vectors X and Y of size dm, $\mathbf{1}_m$ is the m-dimensional vector whose all entries are 1 and $\psi = (\zeta)$.

By replacing the conventional self-attention with the linear self-attention and the efficient self-attention, we obtain the *linear self-attention amortized model* and the *efficient self-attention amortized model*.

Proposition 2. *Self-attention amortized models are symmetric and permutation invariant.*

The proof of Proposition 2 is given in Appendix B.3. The symmetry follows directly from the definition of the self-attention amortized models. The permutation invariance is proved by showing that the self-attention operators combined with average pooling are permutation invariant.

Comparison with Set Transformer. The authors in (Lee et al., 2019b) also proposed a method to guarantee the permutation invariant of sets. There are two main differences between our works and theirs. Firstly, Set Transformer introduced a new attention mechanism and a new Transformer architecture while we only present an approach to apply *any* attention mechanism to preserve the permutation invariance property of amortized models. Secondly, Set Transformer maintains the permutation invariance property by using a learnable multi-head attention as the aggregation scheme. We instead still rely on average pooling, a conventional permutation invariant aggregation scheme, to accumulate features learned by self-attention operations. Nevertheless, our works are orthogonal to Set Transformer, in other words, it is possible to apply techniques in Set Transformer to our attention-based amortized models. We leave this investigation for future work.

Table 1. Reconstruction and transfer learning performance on the ModelNet40 dataset. CD and SW are multiplied by 100.

Method	CD($10^{-2}, \downarrow$)	SW($10^{-2}, \downarrow$)	EMD($\downarrow$)	Acc($\uparrow$)	Time ($\downarrow$)
CD	1.25 ± 0.03	681.20 ± 16.73	653.52 ± 10.03	86.28 ± 0.34	95
EMD	0.40 ± 0.00	94.54 ± 2.90	168.60 ± 1.57	88.45 ± 0.20	208
SW	0.68 ± 0.01	89.61 ± 3.88	191.12 ± 2.88	87.90 ± 0.27	106
Max-SW	0.68 ± 0.01	88.22 ± 1.45	190.23 ± 0.1	87.97 ± 0.14	116
ASW	0.69 ± 0.01	89.42 ± 5.07	192.03 ± 3.09	87.78 ± 0.20	103
v-DSW	$\mathbf{0.67 \pm 0.00}$	85.03 ± 3.31	187.75 ± 2.00	87.83 ± 0.40	633
$\mathcal{L}$-Max-SW	1.06 ± 0.03	121.85 ± 5.77	236.87 ± 3.42	87.70 ± 0.23	$\mathbf{94}$
$\mathcal{G}$-Max-SW	12.11 ± 0.29	851.07 ± 2.11	829.28 ± 5.53	87.49 ± 0.36	97
$\mathcal{N}$-Max-SW	7.38 ± 3.29	618.74 ± 153.87	648.32 ± 117.03	87.43 ± 0.15	96
$\mathcal{L}$v-DSW	0.68 ± 0.00	85.32 ± 0.54	188.32 ± 0.23	87.70 ± 0.34	114
$\mathcal{G}$v-DSW	0.68 ± 0.01	82.77 ± 0.48	187.04 ± 1.11	87.75 ± 0.19	117
$\mathcal{N}$v-DSW	$\mathbf{0.67 \pm 0.00}$	83.47 ± 0.49	186.66 ± 0.81	87.84 ± 0.07	115
$\mathcal{A}$v-DSW	0.67 ± 0.01	83.08 ± 1.22	186.27 ± 0.56	88.05 ± 0.17	230
$\mathcal{EA}$v-DSW	0.68 ± 0.01	82.05 ± 0.40	186.46 ± 0.25	88.07 ± 0.21	125
$\mathcal{LA}$v-DSW	0.68 ± 0.00	$\mathbf{81.03 \pm 0.18}$	$\mathbf{185.26 \pm 0.31}$	$\mathbf{88.28 \pm 0.13}$	123

Table 2. Comparison between amortized models when approximating von Mises Fisher distributional sliced Wasserstein (v-DSW). T denotes the number of projected sub-gradient ascent iterations.

Method	T	Distance ($\uparrow$)	Time ($\downarrow$)
$\mathcal{L}$v-DSW	1	52.73	0.06
$\mathcal{G}$v-DSW	1	50.73	0.07
$\mathcal{N}$v-DSW	1	51.89	0.07
$\mathcal{A}$v-DSW	1	53.07	1.00
$\mathcal{EA}$v-DSW	1	53.17	0.17
$\mathcal{LA}$v-DSW	1	$\mathbf{53.83}$	0.14
v-DSW	1	51.87	0.1
v-DSW	5	51.90	0.33
v-DSW	10	52.65	0.5
v-DSW	50	53.16	2.00
v-DSW	100	54.39	4.00

4. Experiments

To verify the effectiveness of our proposal, we evaluate our methods on the point-cloud reconstruction task and its two downstream tasks including transfer learning and point-cloud generation. Three important questions we want to answer are:

1. Does the sub-optimality issue of amortized Max-SW occur when working with point-clouds and does replacing Max-SW with v-DSW alleviate the problem?

2. Does the proposed amortized distribution projection optimization framework improve the performance over the conventional amortized projection optimization framework and commonly used distances e.g., Chamfer distance, Earth Mover Distance (Wasserstein distance), SW, Max-SW, adaptive SW (ASW) (Nguyen et al., 2021c), and v-DSW?

3. Are self-attention amortized models better than the previous misspecified amortized models in (Nguyen & Ho, 2022a)?

Figure 4. Qualitative results of reconstructing point-clouds in the ShapeNet Core-55 dataset. From top to bottom, the point-clouds are input, SW, Max-SW (T = 50), v-DSW (T = 50), and $\mathcal{LA}$v-DSW respectively.

Table 3. Performance comparison of point-cloud generation on the chair category of ShapeNet. The values of JSD, MMD-CD, and MMD-EMD are multiplied by 100.

Method	JSD $(\times10^{-2},\downarrow)$	MMD $(\times10^{-2},\downarrow)$		COV $(\%,\uparrow)$		1-NNA $(\%,\downarrow)$	
		CD	EMD	CD	EMD	CD	EMD
CD	17.88 ± 1.14	1.12 ± 0.02	17.19 ± 0.36	23.73 ± 1.69	10.83 ± 0.89	98.45 ± 0.10	100.00 ± 0.00
EMD	5.15 ± 1.52	0.61 ± 0.09	10.37 ± 0.61	41.65 ± 2.19	42.54 ± 2.42	87.76 ± 1.46	87.30 ± 1.22
SW	1.56 ± 0.06	0.72 ± 0.02	10.80 ± 0.11	38.55 ± 0.43	45.35 ± 0.48	89.91 ± 1.17	88.28 ± 0.70
Max-SW	1.63 ± 0.32	0.74 ± 0.01	10.84 ± 0.08	40.47 ± 1.04	47.81 ± 0.78	91.46 ± 0.72	89.93 ± 0.86
ASW	1.75 ± 0.38	0.78 ± 0.05	11.27 ± 0.38	38.16 ± 2.15	45.45 ± 1.40	91.21 ± 0.40	89.36 ± 0.40
v-DSW	1.79 ± 0.17	0.72 ± 0.02	10.73 ± 0.20	37.76 ± 0.71	45.49 ± 1.37	90.23 ± 0.13	88.33 ± 0.95
$\mathcal{L}$v-DSW	1.67 ± 0.07	0.77 ± 0.04	11.10 ± 0.33	37.91 ± 1.84	45.64 ± 2.30	90.42 ± 0.53	88.82 ± 0.38
$\mathcal{G}$v-DSW	1.56 ± 0.22	0.75 ± 0.02	10.99 ± 0.11	37.81 ± 1.70	45.69 ± 0.46	90.32 ± 0.38	88.26 ± 0.28
$\mathcal{N}$v-DSW	$\mathbf{1.44 \pm 0.06}$	0.75 ± 0.02	10.95 ± 0.09	38.40 ± 1.34	46.28 ± 2.06	90.15 ± 0.80	88.65 ± 0.82
$\mathcal{EA}$v-DSW	1.73 ± 0.21	$\mathbf{0.71 \pm 0.04}$	$\mathbf{10.70 \pm 0.26}$	40.03 ± 1.28	$\mathbf{48.01 \pm 1.07}$	89.98 ± 0.57	88.55 ± 0.38
$\mathcal{LA}$v-DSW	1.54 ± 0.09	0.72 ± 0.03	10.74 ± 0.35	$\mathbf{40.62 \pm 1.39}$	45.84 ± 1.23	$\mathbf{89.44 \pm 0.28}$	$\mathbf{87.79 \pm 0.37}$

Experiment settings: Our settings[1], which can be found in Appendix C.1, are identical to the setting in the paper of ASW. We compare our methods, amortized v-DSW, with the following loss functions: Chamfer discrepancy (CD), Earth-mover distance (EMD), SW, Max-SW, adaptive SW (ASW),

v-DSW, and amortized Max-SW variants. For amortized models, we consider 6 different ones. The prefix $\mathcal{L}, \mathcal{G}$, and $\mathcal{N}$ denote the linear, generalized linear, and non-linear amortized models in (Nguyen & Ho, 2022a), respectively. $\mathcal{A}, \mathcal{EA}$, and $\mathcal{LA}$ represent self-attention, efficient self-attention, and linear self-attention, respectively. Implementation details for baseline distances and amortized models are given in Appendices C.2 and C.3, respectively. Each experiment was

---

[1]Code for the paper is published at `https://github.com/hsgser/Self-Amortized-DSW`.

run over three different random seeds. We report the average performance along with the standard deviation for each entity. All experiments are run on NVIDIA V100 GPUs.

Comparison with CD and EMD: The main focus of the paper is to compare the new amortized framework with the conventional amortized framework and sliced Wasserstein variants. The results with CD and EMD are provided only for completeness. In addition, we found that there is an unfair comparison between EMD and sliced Wasserstein variants in the ASW's paper. In particular, the EMD loss is normalized by the number of points in a point cloud while SW variants are not. To fix the aforementioned issue, we modified the implementation of the EMD loss by scaling it by the number of points (2048 in this case). As a "perfect" objective, EMD performs better than all SW variants. However, EMD suffers from huge computational costs compared to SW variants

Point-cloud reconstruction: Following ASW (Nguyen et al., 2021c), we measure the reconstruction performance of different autoencoders on the ModelNet40 dataset (Wu et al., 2015) using three discrepancies: Chamfer discrepancy (CD), sliced Wasserstein distance (SW), and EMD. The quantitative results are summarized in Table 1. For each method, we only report the best performing (based on EMD) model among all choices of hyper-parameters. Full quantitative results (including std) can be found in Table 4. Our methods achieve the best performance in all three discrepancies. In contrast, autoencoders with amortized Max-SW losses fail in this scenario due to the sub-optimality and losing metricity issues that we discussed in Section 2.3. In addition, amortized v-DSW losses have smaller standard deviations over 3 runs than v-DSW. Moreover, using amortized optimization reduces the training time compared to the conventional computation using the projected sub-gradient ascent algorithm (e.g. Max-SW and v-DSW). For example, training one iteration of autoencoder using $\mathcal{LA}$v-DSW only takes 123 seconds while using v-DSW costs 633 seconds. In terms of amortized models, attention-based amortized models lead to lower EMD between reconstruction and input. Qualitative results are given in Figure 4, showing the success of our methods in reconstructing 3D point-clouds. Full qualitative results are reported in Figure 6.

Amortization Gaps: To validate the advantage of self-attention amortized models over the previous misspecified amortized models, we compare their effectiveness in approximating v-DSW. We create a dataset by sampling 1000 pairs of point-clouds from the ShapeNet Core-55 dataset. Due to the memory constraint when solving amortized optimization, the dataset is divided into 10 batches of size 100. We compute v-DSW and its amortized versions between all pairs of point-clouds and report their average loss values in Table 2. Compared to previous misspecified amortized mod-

els, attention-based amortized models produce higher losses which are closer to the conventional computation of v-DSW (T = 100). To achieve the same level as efficient/linear self-attention amortized models, one needs to run more than 50 sub-gradient iterations, which is more than 10 times slower.

Transfer learning: We further feed the latent vectors learned by the above autoencoders into a classifier. Following the settings in ASW's paper, we train our classifier for 500 epochs with a batch size of 256. The optimizer is the same as that in the reconstruction experiment. Table 1 illustrates the classification result. Again, we see a boost in accuracy when using self-attention amortized v-DSW.

Point-cloud generation: We also evaluate our methods on the 3D point-cloud generation task. Following (Achlioptas et al., 2018), the chair category of ShapeNet is divided into train/valid/test sets in an 85/5/10 ratio. We train each autoencoder on the train set for 100 epochs and evaluate on the valid set. The generator is then trained to generate latent codes learned by the autoencoder, same as (Achlioptas et al., 2018). For evaluation, the same set of metrics in (Yang et al., 2019a) is used. The quantitative results of the test set are given in Table 3. Our methods yield the best performance in all metrics. In addition, attention-based amortized models lead to higher performance than previous amortized models in all metrics except for JSD. Full quantitative results are reported in Table 9.

5. Conclusion

We have proposed a self-attention amortized distributional projection optimization framework which uses a self-attention amortized model to predict the best discriminative distribution over projecting direction for each pair of probability measures. The efficient self-attention mechanism helps to inject the geometric inductive biases which are permutation invariance and symmetry into the amortized model while remaining fast computation. Furthermore, the amortized distribution projection optimization framework guarantees the metricity for all pairs of probability measures while the amortization gap still exists. On the experimental side, we compare the new proposed framework to the conventional amortized projection optimization framework and other widely-used distances in the point-cloud reconstruction application and its two downstream tasks including transfer learning and point-cloud generation to show the superior performance of the proposed framework.

Acknowledgements

References

Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pp. 40–49. PMLR, 2018.

Alvarez-Melis, D. and Fusi, N. Geometric dataset distances via optimal transport. *Advances in Neural Information Processing Systems*, 33:21428–21439, 2020.

Amos, B. Tutorial on amortized optimization for learning to optimize over continuous domains. *arXiv preprint arXiv:2202.00665*, 2022.

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

Barrow, H. G., Tenenbaum, J. M., Bolles, R. C., and Wolf, H. C. Parametric correspondence and chamfer matching: Two new techniques for image matching. Technical report, SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1977.

Bhushan Damodaran, B., Kellenberger, B., Flamary, R., Tuia, D., and Courty, N. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 447–463, 2018.

Bonneel, N., Rabin, J., Peyré, G., and Pfister, H. Sliced and Radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 1(51):22–45, 2015.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *European conference on computer vision*, pp. 213–229. Springer, 2020.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

Christensen, R. *Plane answers to complex questions*, volume 35. Springer, 2002.

Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9): 1853–1865, 2016.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, pp. 2292–2300, 2013.

Damodaran, B. B., Kellenberger, B., Flamary, R., Tuia, D., and Courty, N. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 447–463, 2018.

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., and Tomczak, J. M. Hyperspherical variational auto-encoders. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pp. 856–865. Association For Uncertainty in Artificial Intelligence (AUAI), 2018a.

Davidson, T. R., Falorsi, L., De Cao, N., and Tomczak, T. K. J. M. Hyperspherical variational auto-encoders. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018b.

Deshpande, I., Zhang, Z., and Schwing, A. G. Generative modeling using the sliced Wasserstein distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3483–3491, 2018.

Deshpande, I., Hu, Y.-T., Sun, R., Pyrros, A., Siddiqui, N., Koyejo, S., Zhao, Z., Forsyth, D., and Schwing, A. G. Max-sliced Wasserstein distance and its use for GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10648–10656, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

Genevay, A., Peyré, G., and Cuturi, M. Learning generative models with Sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pp. 1608–1617. PMLR, 2018.

Guo, M.-H., Cai, J.-X., Liu, Z.-N., Mu, T.-J., Martin, R. R., and Hu, S.-M. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, Apr 2021. ISSN 2096-0662. doi: 10.1007/s41095-021-0229-5. URL http://dx.doi.org/10.1007/s41095-021-0229-5.

Ho, N., Nguyen, X., Yurochkin, M., Bui, H. H., Huynh, V., and Phung, D. Multilevel clustering via Wasserstein means. In *International Conference on Machine Learning*, pp. 1501–1509, 2017.

Huynh, V., Zhao, H., and Phung, D. Otlda: A geometry-aware optimal transport approach for topic modeling. *Advances in Neural Information Processing Systems*, 33: 18573–18582, 2020.

Jupp, P. E. and Mardia, K. V. Maximum likelihood estimators for the matrix von Mises-Fisher and bingham distributions. *The Annals of Statistics*, 7(3):599–606, 1979.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kolouri, S., Pope, P. E., Martin, C. E., and Rohde, G. K. Sliced Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018.

Lee, C.-Y., Batra, T., Baig, M. H., and Ulbricht, D. Sliced Wasserstein discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10285–10295, 2019a.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019b.

Lee, Y., Kim, S., Choi, J., and Park, F. A statistical manifold framework for point cloud data. In *International Conference on Machine Learning*, pp. 12378–12402. PMLR, 2022.

Li, R., Su, J., Duan, C., and Zheng, S. Linear attention mechanism: An efficient attention for semantic segmentation. *arXiv preprint arXiv:2007.14902*, 2020.

Li, S., Raj, D., Lu, X., Shen, P., Kawahara, T., and Kawai, H. Improving Transformer-Based Speech Recognition Systems with Compressed Structure and Speech Attributes Augmentation. In *Proc. Interspeech 2019*, pp. 4400–4404, 2019. doi: 10.21437/Interspeech.2019-2112. URL http://dx.doi.org/10.21437/Interspeech.2019-2112.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Naderializadeh, N., Comer, J., Andrews, R., Hoffmann, H., and Kolouri, S. Pooling by sliced-Wasserstein embedding. *Advances in Neural Information Processing Systems*, 34, 2021.

Nadjahi, K., De Bortoli, V., Durmus, A., Badeau, R., and Şimşekli, U. Approximate Bayesian computation with the sliced-Wasserstein distance. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5470–5474. IEEE, 2020.

Nguyen, K. and Ho, N. Amortized projection optimization for sliced Wasserstein generative models. *Advances in Neural Information Processing Systems*, 2022a.

Nguyen, K. and Ho, N. Revisiting sliced Wasserstein on images: From vectorization to convolution. *Advances in Neural Information Processing Systems*, 2022b.

Nguyen, K., Ho, N., Pham, T., and Bui, H. Distributional sliced-Wasserstein and applications to generative modeling. In *International Conference on Learning Representations*, 2021a.

Nguyen, K., Nguyen, S., Ho, N., Pham, T., and Bui, H. Improving relational regularized autoencoders with spherical sliced fused Gromov Wasserstein. In *International Conference on Learning Representations*, 2021b.

Nguyen, K., Ren, T., Nguyen, H., Rout, L., Nguyen, T. M., and Ho, N. Hierarchical sliced wasserstein distance. In *The Eleventh International Conference on Learning Representations*, 2023.

Nguyen, T., Pham, Q.-H., Le, T., Pham, T., Ho, N., and Hua, B.-S. Point-set distances for learning representations of 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021c.

Nietert, S., Sadhu, R., Goldfeld, Z., and Kato, K. Statistical, robustness, and computational guarantees for sliced wasserstein distances. *Advances in Neural Information Processing Systems*, 2022.

11

Peyré, G. and Cuturi, M. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Pham, Q.-H., Uy, M. A., Hua, B.-S., Nguyen, D. T., Roig, G., and Yeung, S.-K. LCD: Learned cross-domain descriptors for 2D-3D matching. In *AAAI Conference on Artificial Intelligence*, 2020.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Shen, Z., Zhang, M., Zhao, H., Yi, S., and Li, H. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 3531–3539, 2021.

Shu, R. Amortized optimization `http://ruishu.io/2017/11/07/amortized-optimization/`. *Personal Blog*, 2017.

Sra, S. Directional statistics in machine learning: a brief review. *arXiv preprint arXiv:1605.00316*, 2016.

Sun, C., Myers, A., Vondrick, C., Murphy, K., and Schmid, C. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7464–7473, 2019.

Temme, N. M. *Special functions: An introduction to the classical functions of mathematical physics*. John Wiley & Sons, 2011.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Villani, C. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Vincent-Cuaz, C., Flamary, R., Corneli, M., Vayer, T., and Courty, N. Template based graph neural network with optimal transport distances. *Advances in Neural Information Processing Systems*, 2022.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020a.

Wang, Y., Mohamed, A., Le, D., Liu, C., Xiao, A., Mahadeokar, J., Huang, H., Tjandra, A., Zhang, X., Zhang, F., et al. Transformer-based acoustic modeling for hybrid speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6874–6878. IEEE, 2020b.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

Yang, G., Huang, X., Hao, Z., Liu, M.-Y., Belongie, S., and Hariharan, B. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4541–4550, 2019a.

Yang, J., Zhang, Q., Ni, B., Li, L., Liu, J., Zhou, M., and Tian, Q. Modeling point clouds with self-attention and gumbel subset sampling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3323–3332, 2019b.

Yi, M. and Liu, S. Sliced Wasserstein variational inference. In *Fourth Symposium on Advances in Approximate Bayesian Inference*, 2021.

Zhao, H., Jiang, L., Jia, J., Torr, P. H., and Koltun, V. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16259–16268, 2021.

Supplement to "Self-Attention Amortized Distributional Projection Optimization for Sliced Wasserstein Point-Clouds Reconstruction"

In this supplementary, we first provide some additional materials in Appendix A including definitions of generalized linear amortized models and non-linear amortized models in Appendix A.1, the detail of computing von Mises-Fisher distributional sliced Wasserstein in Appendix A.2, and training algorithms for autoencoders in Appendix A.3. Next, we collect skipped proofs in the main text in Appendix B. After that, we discuss the experimental settings of our experiments in Appendix C. Finally, we present additional experimental results in Appendix D.

A. Additional Materials

A.1. Amortized models

We now review the generalized linear amortized model and the non-linear amortized model (Nguyen & Ho, 2022a).

Definition 4. Given $X, Y \in \mathbb{R}^{dm}$, the *generalized linear amortized model* is defined as:

$$f_\psi(X, Y) := \frac{g_{\psi_1}(X)'w_1 + g_{\psi_1}(Y)'w_2}{||g_{\psi_1}(X)'w_1 + g_{\psi_1}(Y)'w_2||_2^2}, \tag{17}$$

where X' and Y' are matrices of size $d \times m$ that are reshaped from the concatenated vectors X and Y of size dm, $w_1, w_2 \in \mathbb{R}^m$, $w_0 \in \mathbb{R}^d$, $\psi_1 \in \Psi_1$, $g_{\psi_1} : \mathbb{R}^{dm} \to \mathbb{R}^{dm}$, $\psi = (w_0, w_1, w_2, \psi_1)$, and $g_{\psi_1}(X) = (x_1', \ldots, x_m')$ and $g_{\psi_1}(Y) = (y_1', \ldots, y_m')$. To specify, we let $g_{\psi_1}(X) = (W_2\sigma(W_1 x_1) + b_0, \ldots, W_2\sigma(W_1 x_m) + b_0)$, where $\sigma(\cdot)$ is the Sigmoid function, $W_1 \in \mathbb{R}^{d \times d}$, $W_2 \in \mathbb{R}^{d \times d}$, and $b_0 \in \mathbb{R}^d$.

Definition 5. Given $X, Y \in \mathbb{R}^{dm}$, the *non-linear amortized model* is defined as:

$$f_\psi(X, Y) := \frac{h_{\psi_2}(X'w_1 + Y'w_2)}{||h_{\psi_2}(X'w_1 + Y'w_2)||_2^2}, \tag{18}$$

where X' and Y' are matrices of size $d \times m$ that are reshaped from the concatenated vectors X and Y of size dm, $w_1, w_2 \in \mathbb{R}^m$, $\psi_2 \in \Psi_2$, $h_{\psi_2} : \mathbb{R}^d \to \mathbb{R}^d$, $\psi = (w_1, w_2, \psi_2)$, and $h_{\psi_2}(x) = W_4\sigma(W_3 x)) + b_0$ where $\sigma(\cdot)$ is the Sigmoid function.

A.2. Von Mises-Fisher distributional sliced Wasserstein distance

We first start with the definition of von Mises Fisher (vMF) distribution. The von Mises–Fisher distribution (*vMF*)(Jupp & Mardia, 1979) is a probability distribution on the unit hypersphere $\mathbb{S}^{d-1}$ with the density function is :

$$f(x|\epsilon, \kappa) := C_d(\kappa)\exp(\kappa\epsilon^\top x), \tag{19}$$

where $\epsilon \in \mathbb{S}^{d-1}$ is the location vector, $\kappa \geq 0$ is the concentration parameter, and $C_d(\kappa) := \frac{\kappa^{d/2-1}}{(2\pi)^{d/2}I_{d/2-1}(\kappa)}$ is the normalization constant. Here, I_v is the modified Bessel function of the first kind at order v (Temme, 2011).

The vMF distribution is a continuous distribution, its mass concentrates around the mean ϵ, and its density decrease when x goes away from ϵ. When $\kappa \to 0$, vMF converges in distribution to $\mathcal{U}(\mathbb{S}^{d-1})$, and when $\kappa \to \infty$, vMF converges in distribution to the Dirac distribution centered at ϵ (Sra, 2016).

Reparameterized Rejection Sampling: The sampling process of vMF distribution is based on the rejection sampling procedure. We review the sampling process in Algorithm 1 (Davidson et al., 2018a; Nguyen et al., 2021b). The algorithm performs the reparameterization for the proposal distribution. We now derive the gradient estimator for $\nabla_\epsilon \mathbb{E}_{\text{vMF}(\theta|\epsilon, \kappa)}[f(\theta)]$ for a general function $f(\theta)$ to find the maxima ϵ^* in the optimization problem $\max_{\epsilon \in \mathbb{S}^{d-1}} \mathbb{E}_{\text{vMF}(\theta|\epsilon, \kappa)}[f(\theta)]$.

In $d > 0$ dimension, let (ϵ, κ) be the parameters of vMF distribution. We denotes $b = \frac{-2\kappa + \sqrt{4\kappa^2 + (d-1)^2}}{d-1}$, two conditional distributions: $g(\omega \mid \kappa) = \frac{2(\pi^{d/2})}{\Gamma(d/2)}\mathcal{C}_d(\kappa)\frac{\exp(\omega\kappa)(1-\omega^2)^{\frac{1}{2}(d-3)}}{\text{Beta}(\frac{1}{2}, \frac{1}{2}(d-1))}$, $r(\omega|\kappa) = \frac{2b^{1/2(d-1)}}{\text{Beta}(\frac{1}{2}(d-1), \frac{1}{2}(d-1))}\frac{(1-\omega^2)^{1/2(d-3)}}{[(1+b)-(1-b)\omega]^{d-1}}$, distribution $s(\psi) := \text{Beta}(\frac{1}{2}(d-1), \frac{1}{2}(d-1))$, function $h(\psi, \kappa) = \frac{1-(1+b)\psi}{1-(1-b)\psi}$, distributions $\pi_1(\psi|\kappa) = s(\psi)\frac{g(h(\psi,\kappa)|\kappa)}{r(h(\psi,\kappa)|\kappa)}$, $\pi_2(v) := \mathcal{U}(\mathbb{S}^{d-2})$, and function $T(\omega, v, \epsilon) = \left(I - 2\frac{e_1 - \epsilon}{||e_1 - \epsilon||_2}\frac{e_1 - \epsilon}{||e_1 - \epsilon||_2}^\top\right)(\omega, \sqrt{1-\omega^2}v^\top)^\top := \theta$.

Algorithm 1 Sampling from vMF distribution

Input: location ϵ, concentration κ, dimension d, unit vector $e_1 = (1, 0, .., 0)$
Draw $v \sim \mathcal{U}(\mathbb{S}^{d-2})$
$b \leftarrow \frac{-2\kappa + \sqrt{4\kappa^2 + (d-1)^2}}{d-1}, a \leftarrow \frac{(d-1) + 2\kappa + \sqrt{4\kappa^2 + (d-1)^2}}{4}, m \leftarrow \frac{4ab}{(1+b)} - (d-1)\log(d-1)$
repeat
    Draw $\psi \sim \text{Beta}\left(\frac{1}{2}(d-1), \frac{1}{2}(d-1)\right)$
    $\omega \leftarrow h(\psi, \kappa) = \frac{1 - (1+b)\psi}{1 - (1-b)\psi}$
    $t \leftarrow \frac{2ab}{1 - (1-b)\psi}$
    Draw $u \sim \mathcal{U}([0,1])$
until $(d-1)\log(t) - t + m \geq \log(u)$
$h_1 \leftarrow (\omega, \sqrt{1 - \omega^2} v^\top)^\top$
$\epsilon' \leftarrow e_1 - \epsilon$
$u = \frac{\epsilon'}{||\epsilon'||_2}$
$U = I - 2uu^\top$
Output: Uh_1

We can obtain the gradient estimator by the following Lemma 2 in (Davidson et al., 2018b):

$$\nabla_\epsilon \mathbb{E}_{\text{vMF}(\theta|\epsilon,\kappa)}\big[f(\theta)\big] = \nabla_\epsilon \mathbb{E}_{(\psi,v)\sim\pi_1(\psi|\kappa)\pi_2(v)}\Big[f\big(T(h(\psi,\kappa), v, \epsilon)\big)\Big] = \mathbb{E}_{(\psi,v)\sim\pi_1(\psi|\kappa)\pi_2(v)}\Big[\nabla_\epsilon f\big(T(h(\psi,\kappa), v, \epsilon)\big)\Big].$$

In v-DSW case, we have $f(\theta) = \mathrm{W}_p^p(\theta\sharp\mu, \theta\sharp\nu)$. Therefore, we have:

$$\nabla_\epsilon \mathbb{E}_{\text{vMF}(\theta|\epsilon,\kappa)}\big[\mathrm{W}_p^p(\theta\sharp\mu, \theta\sharp\nu)\big] = \mathbb{E}_{(\psi,v)\sim\pi_1(\psi|\kappa)\pi_2(v)}\Big[\nabla_\epsilon \mathrm{W}_p^p(f\big(T(h(\psi,\kappa), v, \epsilon)\sharp\mu, f\big(T(h(\psi,\kappa), v, \epsilon)\sharp\nu)\big)\Big].$$

Then we can get a gradient estimator by using Monte-Carlo estimation scheme:

$$\nabla_\epsilon \mathbb{E}_{\text{vMF}(\theta|\epsilon,\kappa)}\big[\mathrm{W}_p^p(\theta\sharp\mu, \theta\sharp\nu)\big] \approx \frac{1}{L}\sum_{i=1}^{L}\Big[\nabla_\epsilon \mathrm{W}_p^p(f\big(T(h(\psi_i,\kappa), v_i, \epsilon)\sharp\mu, f\big(T(h(\psi_i,\kappa), v_i, \epsilon)\sharp\nu)\big)\Big],$$

where $\{\psi_i\}_{i=1}^{L} \sim \pi_1(\psi|\kappa)$ i.i.d, $\{v_i\}_{i=1}^{L} \sim \pi_2(v)$ i.i.d, and L is the number of projections. Sampling from $\pi_1(\psi|\kappa)$ is equivalent to the acceptance-rejection scheme in vMF sampling procedure, sampling $\pi_2(v)$ is directly from $\mathcal{U}(\mathbb{S}^{d-2})$. It is worth noting that the gradient estimator for $\nabla_\kappa \mathbb{E}_{\text{vMF}(\theta|\epsilon,\kappa)}\big[f(\theta)\big]$ can be derived by using the log-derivative trick, however, we do not need it here since we do not optimize for κ in v-DSW.

A.3. Training algorithms

Training point-cloud autoencoder with Max-SW: We present the algorithm of training autoencoder with Max-SW in Algorithm 2. The algorithm contains a nested loop: one is for training the autoencoder, one is for finding the max projecting direction for Max-SW.

Training point-cloud autoencoder with amortized projection optimization: We present the training algorithm for point-cloud autoencoder with amortized projection optimization in Algorithm 3. With amortized optimization, the inner loop for finding the max projecting direction is removed.

Training point-cloud autoencoder with v-DSW: We present the algorithm of training autoencoder with v-DSW in Algorithm 4. The algorithm contains a nested loop: one is for training the autoencoder, one is for finding the best distribution over projecting directions for v-DSW.

Training point-cloud autoencoder with amortized distributonal projection optimization: We present the training algorithm for point-cloud autoencoder with amortized distributional projection optimization in Algorithm 5. With amortized distributional optimization, the inner loop for finding the best distribution over projecting directions is removed.

Algorithm 2 Training point-cloud autoencoder with max sliced Wasserstein distance

---

Input: Point-cloud distribution $p(X)$, learning rate η, slice learning rate η_s, model maximum number of iterations $\mathcal{T}$, slice maximum number of iterations T, mini-batch size k.

Initialization: Initialize the encoder f_ϕ and the decoder g_γ

while ϕ, γ not converge or reach $\mathcal{T}$ **do**

    Sample a mini-batch $X_1, \ldots, X_k$ i.i.d from $p(X)$

    $\nabla_\phi = 0, \nabla_\gamma = 0$

    for $i = 1$ to k **do**

        Initialize θ

        while θ not converge or reach T **do**

            $\theta = \theta + \eta_s \cdot \nabla_\theta \mathrm{W}_p(\theta \sharp P_{X_i}, \theta \sharp P_{g_\gamma(f_\phi(X_i))})$ # Other update rules can be used

            $\theta = \frac{\theta}{||\theta||_2}$ #Project back to the unit-hypersphere $\mathbb{S}^{d-1}$

        end while

        $\nabla_\phi = \nabla_\phi + \frac{1}{k} \nabla_\phi \mathrm{W}_p(\theta \sharp P_{X_i}, \theta \sharp P_{g_\gamma(f_\phi(X_i))})$

        $\nabla_\gamma = \nabla_\gamma + \frac{1}{k} \nabla_\gamma \mathrm{W}_p(\theta \sharp P_{X_i}, \theta \sharp P_{g_\gamma(f_\phi(X_i))})$

    end for

    $\phi = \phi - \eta \cdot \nabla_\phi$ # Other update rules can be used

    $\gamma = \gamma - \eta \cdot \nabla_\gamma$ # Other update rules can be used

end while

Return: ϕ, γ

Algorithm 3 Training point-cloud autoencoder with amortized projection optimization

---

Input: Point-cloud distribution $p(X)$, learning rate η, slice learning rate η_s, model maximum number of iterations $\mathcal{T}$, mini-batch size k.

Initialization: Initialize the encoder f_ϕ, the decoder g_γ, and the amortized model a_ψ

while ϕ, γ, ψ not converge or reach $\mathcal{T}$ **do**

    Sample a mini-batch $X_1, \ldots, X_k$ i.i.d from $p(X)$

    $\nabla_\phi = 0, \nabla_\gamma = 0, \nabla_\psi = 0$

    for $i = 1$ to k **do**

        $\theta_{\psi,\gamma,\phi} = a_\psi(X_i, g_\gamma(f_\phi(X_i)))$

        $\nabla_\psi = \nabla_\psi + \frac{1}{k} \nabla_\psi \mathrm{W}_p(\theta_{\psi,\gamma,\phi} \sharp P_{X_i}, \theta_{\psi,\gamma,\phi} \sharp P_{g_\gamma(f_\phi(X_i))})$

        $\nabla_\phi = \nabla_\phi + \frac{1}{k} \nabla_\phi \mathrm{W}_p(\theta_{\psi,\gamma,\phi} \sharp P_{X_i}, \theta_{\psi,\gamma,\phi} \sharp P_{g_\gamma(f_\phi(X_i))})$

        $\nabla_\gamma = \nabla_\gamma + \frac{1}{k} \nabla_\gamma \mathrm{W}_p(\theta_{\psi,\gamma,\phi} \sharp P_{X_i}, \theta_{\psi,\gamma,\phi} \sharp P_{g_\gamma(f_\phi(X_i))})$

    end for

    $\psi = \psi + \eta_s \cdot \nabla_\psi$ # Other update rules can be used

    $\phi = \phi - \eta \cdot \nabla_\phi$ # Other update rules can be used

    $\gamma = \gamma - \eta \cdot \nabla_\gamma$ # Other update rules can be used

end while

Return: ϕ, γ

B. Proofs

B.1. Proof for Proposition 1

We first recall the definition of the projected one-dimensional Wasserstein between two probability measures μ and ν: $\mathrm{PW}_p(\mu, \nu; \hat{\theta}) = \mathrm{W}_p(\hat{\theta} \sharp \mu, \hat{\theta} \sharp \nu)$ for $\hat{\theta} \neq \arg\max_{\theta \in \mathbb{S}^{d-1}} \mathrm{W}_p(\theta \sharp \mu, \theta \sharp \nu)$.

Non-negativity and Symmetry: Due to the non-negativity and symmetry of the Wasserstein distance, the non-negativity and symmetry of the projected Wasserstein follow directly from its definition.

15

Algorithm 4 Training point-cloud autoencoder with von-Mises Fisher distributional sliced Wasserstein distance

---

Input: Point-cloud distribution $p(X)$, learning rate η, slice learning rate η_s, model maximum number of iterations $\mathcal{T}$, slice maximum number of iterations T, mini-batch size k, the number of projections L, and the concentration hyperparameter κ.

Initialization: Initialize the encoder f_ϕ and the decoder g_γ

while ϕ, γ not converge or reach $\mathcal{T}$ **do**

    Sample a mini-batch $X_1, \ldots, X_k$ i.i.d from $p(X)$

    $\nabla_\phi = 0, \nabla_\gamma = 0$

    for $i = 1$ to k **do**

        Initialize ϵ

        while ϵ not converge or reach T **do**

            Sample $\theta_1^\epsilon, \ldots, \theta_L^\epsilon$ i.i.d from vMF(ϵ, κ) via the reparameterized acceptance-rejection sampling in Algorithm 1

            $\epsilon = \epsilon + \eta_s \cdot \frac{1}{L} \sum_{l=1}^{L} \nabla_\epsilon W_p(\theta_l^\epsilon \sharp P_{X_i}, \theta_l^\epsilon \sharp P_{g_\gamma(f_\phi(X_i))})$ # Other update rules can be used

            $\epsilon = \frac{\epsilon}{||\epsilon||_2}$ #Project back to the unit-hypersphere $\mathbb{S}^{d-1}$

        end while

        Sample $\theta_1^\epsilon, \ldots, \theta_L^\epsilon$ i.i.d from vMF(ϵ, κ) via the reparameterized acceptance-rejection sampling in Algorithm 1.

        $\nabla_\phi = \nabla_\phi + \frac{1}{k}\frac{1}{L} \sum_{i=l}^{L} \nabla_\phi W_p(\theta_l^\epsilon \sharp P_{X_i}, \theta_l^\epsilon \sharp P_{g_\gamma(f_\phi(X_i))})$

        $\nabla_\gamma = \nabla_\gamma + \frac{1}{k}\frac{1}{L} \sum_{i=l}^{L} \nabla_\gamma W_p(\theta_l^\epsilon \sharp P_{X_i}, \theta_l^\epsilon \sharp P_{g_\gamma(f_\phi(X_i))})$

    end for

    $\phi = \phi - \eta \cdot \nabla_\phi$ # Other update rules can be used

    $\gamma = \gamma - \eta \cdot \nabla_\gamma$ # Other update rules can be used

end while

Return: ϕ, γ

Algorithm 5 Training point-cloud autoencoder with amortized projection optimization

---

Input: Point-cloud distribution $p(X)$, learning rate η, slice learning rate η_s, model maximum number of iterations $\mathcal{T}$, mini-batch size k.

Initialization: Initialize the encoder f_ϕ, the decoder g_γ, and the amortized model a_ψ

while ϕ, γ, ψ not converge or reach $\mathcal{T}$ **do**

    Sample a mini-batch $X_1, \ldots, X_k$ i.i.d from $p(X)$

    $\nabla_\phi = 0, \nabla_\gamma = 0, \nabla_\psi = 0$

    for $i = 1$ to k **do**

        $\epsilon_{\psi,\gamma,\phi} = a_\psi(X_i, g_\gamma(f_\phi(X_i)))$

        Sample $\theta_1^{\psi,\gamma,\phi}, \ldots, \theta_L^{\psi,\gamma,\phi}$ i.i.d from vMF$(\epsilon_{\psi,\gamma,\phi}, \kappa)$ via the reparameterized acceptance-rejection sampling in Algorithm 1

        $\nabla_\psi = \nabla_\psi + \frac{1}{k}\frac{1}{L} \sum_{i=l}^{L} \nabla_\psi W_p(\theta_l^{\psi,\gamma,\phi} \sharp P_{X_i}, \theta_l^{\psi,\gamma,\phi} \sharp P_{g_\gamma(f_\phi(X_i))})$

        $\nabla_\phi = \nabla_\phi + \frac{1}{k}\frac{1}{L} \sum_{i=l}^{L} \nabla_\phi W_p(\theta_l^{\psi,\gamma,\phi} \sharp P_{X_i}, \theta_l^{\psi,\gamma,\phi} \sharp P_{g_\gamma(f_\phi(X_i))})$

        $\nabla_\gamma = \nabla_\gamma + \frac{1}{k}\frac{1}{L} \sum_{i=l}^{L} \nabla_\gamma W_p(\theta_l^{\psi,\gamma,\phi} \sharp P_{X_i}, \theta_l^{\psi,\gamma,\phi} \sharp P_{g_\gamma(f_\phi(X_i))})$

    end for

    $\psi = \psi + \eta_s \cdot \nabla_\psi$ # Other update rules can be used

    $\phi = \phi - \eta \cdot \nabla_\phi$ # Other update rules can be used

    $\gamma = \gamma - \eta \cdot \nabla_\gamma$ # Other update rules can be used

end while

Return: ϕ, γ

Triangle inequality: For any three probability measures $\mu_1, \mu_2, \mu_3 \in \mathcal{P}_p(\mathbb{R}^d)$, we have:

$$\begin{aligned} \mathrm{PW}_p(\mu_1, \mu_3; \hat{\theta}) &= \mathrm{W}_p(\hat{\theta}\sharp\mu_1, \hat{\theta}\sharp\mu_3) \\ &\leq \mathrm{W}_p(\hat{\theta}\sharp\mu_1, \hat{\theta}\sharp\mu_2) + \mathrm{W}_p(\hat{\theta}\sharp\mu_2, \hat{\theta}\sharp\mu_3) \\ &= \mathrm{PW}_p(\mu_1, \mu_2; \hat{\theta}) + \mathrm{PW}_p(\mu_2, \mu_3; \hat{\theta}), \end{aligned}$$

where the first inequality is due to the triangle inequality of the Wasserstein distance.

Identity: If $\mu = \nu$, we have $\text{PW}_p(\mu, \nu; \hat{\theta}) = 0$ due to the identity of the Wasserstein distance. However, if $\text{PW}_p(\mu, \nu; \hat{\theta}) = 0$, there exists $\theta' \in \mathbb{S}^{d-1}$ such that $0 = \text{PW}_p(\mu, \nu; \hat{\theta}) < \text{PW}_p(\mu, \nu; \theta')$. Let $\mathcal{F}[\gamma](w) = \int_{\mathbb{R}^{d'}} e^{-i\langle w, x \rangle} d\gamma(x)$ be the Fourier transform of $\gamma \in \mathcal{P}(\mathbb{R}^{d'})$, for any $t \in \mathbb{R}$, we have

$$\mathcal{F}[\mu](t\theta') = \int_{\mathbb{R}^d} e^{-it\langle \theta', x \rangle} d\mu(x) = \int_{\mathbb{R}} e^{-itz} d\theta' \sharp \mu(z) = \mathcal{F}[\theta' \sharp \mu](t)$$

$$\neq \mathcal{F}[\theta' \sharp \nu](t) = \int_{\mathbb{R}} e^{-itz} d\theta' \sharp \nu(z) = \int_{\mathbb{R}^d} e^{-it\langle \theta', x \rangle} d\nu(x) = \mathcal{F}[\nu](t\theta').$$

Therefore, we have $\mu \neq \nu$. We complete the proof.

B.2. Proof for Theorem 1

We first start with proving the metricity of the *non-optimal* von Mises Fisher distributional sliced Wasserstein distance (v-DSW). For any two probability measures $\mu, \nu \in \mathcal{P}_p(\mathbb{R}^d)$, the *non-optimal* von Mises Fisher distributional sliced Wasserstein distance (v-DSW) is defined as follow:

$$\text{v-DSW}_p(\mu, \nu; \epsilon, \kappa) = \left(\mathbb{E}_{\theta \sim \text{vMF}(\epsilon, \kappa)} \text{W}_p^p(\theta \sharp \mu, \theta \sharp \nu) \right)^{\frac{1}{p}},$$

where $\epsilon \in \mathbb{S}^{d-1}$ and $0 < \kappa < \infty$.

Lemma 1. *For any $\epsilon \in \mathbb{S}^{d-1}$ and $\kappa < \infty$, v-DSW$_p(\cdot, \cdot; \epsilon, \kappa)$ is a valid metric on the space of probability measures.*

Proof. We now prove that v-DSW satisfies non-negativity, symmetry, triangle inequality, and identity.

Non-negativity and Symmetry: The non-negativity and symmetry of v-DSW follow directly the non-negativity and symmetry of the Wasserstein distance.

Triangle inequality: For any three probability measures $\mu_1, \mu_2, \mu_3 \in \mathcal{P}_p(\mathbb{R}^d)$, we have

$$\text{v-DSW}_p(\mu_1, \mu_3; \epsilon, \kappa) = \left(\mathbb{E}_{\theta \sim \text{vMF}(\epsilon, \kappa)} \text{W}_p^p(\theta \sharp \mu_1, \theta \sharp \mu_3) \right)^{\frac{1}{p}}$$

$$\leq \left(\mathbb{E}_{\theta \sim \text{vMF}(\epsilon, \kappa)} \left[\text{W}_p(\theta \sharp \mu_1, \theta \sharp \mu_2) + \text{W}_p(\theta \sharp \mu_2, \theta \sharp \mu_3) \right]^p \right)^{\frac{1}{p}}$$

$$\leq \left(\mathbb{E}_{\theta \sim \text{vMF}(\epsilon, \kappa)} \text{W}_p^p(\theta \sharp \mu_1, \theta \sharp \mu_2) \right)^{\frac{1}{p}} + \left(\mathbb{E}_{\theta \sim \text{vMF}(\epsilon, \kappa)} \text{W}_p^p(\theta \sharp \mu_2, \theta \sharp \mu_3) \right)^{\frac{1}{p}}$$

$$= \text{v-DSW}_p(\mu_1, \mu_2; \epsilon, \kappa) + \text{v-DSW}_p(\mu_2, \mu_3; \epsilon, \kappa)$$

Identity: From the definition, if $\mu = \nu$, we obtain v-DSW$_p(\mu, \nu; \epsilon, \kappa) = 0$. Now, we need to show that if v-DSW$_p(\mu, \nu; \epsilon, \kappa) = 0$, then $\mu = \nu$.

If v-DSW$_p(\mu, \nu; \epsilon, \kappa) = 0$, we have $\left(\mathbb{E}_{\theta \sim \text{vMF}(\epsilon, \kappa)} \text{W}_p^p(\theta \sharp \mu, \theta \sharp \nu) \right)^{\frac{1}{p}} = 0$ which implies $\mathbb{E}_{\theta \sim \text{vMF}(\epsilon, \kappa)} \text{W}_p^p(\theta \sharp \mu, \theta \sharp \nu) = 0$. Therefore, $\text{W}_p(\theta \sharp \mu, \theta \sharp \nu) = 0$ for vMF(ϵ, κ) almost surely $\theta \in \mathbb{S}^{d-1}$. Using the identity property of the Wasserstein distance, we obtain $\theta \sharp \mu = \theta \sharp \nu$ for vMF(ϵ, κ) almost surely $\theta \in \mathbb{S}^{d-1}$. Since vMF$(\epsilon, \kappa)$ with $0 < \kappa < \infty$ has the supports on all $\mathbb{S}^{d-1}$, for any $t \in \mathbb{R}$ and $\theta \in \mathbb{S}^{d-1}$, we have:

$$\mathcal{F}[\mu](t\theta) = \int_{\mathbb{R}^d} e^{-it\langle \theta, x \rangle} d\mu(x) = \int_{\mathbb{R}} e^{-itz} d\theta \sharp \mu(z) = \mathcal{F}[\theta \sharp \mu](t)$$

$$= \mathcal{F}[\theta \sharp \nu](t) = \int_{\mathbb{R}} e^{-itz} d\theta \sharp \nu(z) = \int_{\mathbb{R}^d} e^{-it\langle \theta, x \rangle} d\nu(x) = \mathcal{F}[\nu](t\theta),$$

where $\mathcal{F}[\gamma](w) = \int_{\mathbb{R}^{d'}} e^{-i\langle w, x \rangle} d\gamma(x)$ denotes the Fourier transform of $\gamma \in \mathcal{P}(\mathbb{R}^{d'})$. We then obtain $\mu = \nu$ by the injectivity of the Fourier transform. We complete the proof. $\square$

By abuse of notation, we denote v-DSW$(X, Y; \epsilon, \kappa) = $ v-DSW$(P_X, P_Y; \epsilon, \kappa)$ for $X, Y \in \mathcal{X}$ are two point-clouds, $P_X = \frac{1}{m} \sum_{i=1}^m \delta_{x_i}$, and $P_Y = \frac{1}{m} \sum_{i=1}^m \delta_{y_i}$. We cast the v-DSW from a metric on the space of probability measures to the space of point-clouds $\mathcal{X}$.

Corollary 1. *For any $\epsilon \in \mathbb{S}^{d-1}$ and $\kappa < \infty$, v-DSW$_p(\cdot, ; \epsilon, \kappa)$ is a valid metric on the space of point-clouds $\mathcal{X}$.*

Proof. Since $P_X, P_Y \in \mathcal{P}_p(\mathbb{R}^d)$, the non-negativity, symmetry, triangle inequality, and identity properties follow directly from Lemma 1. We now only need to show that v-DSW is invariant to permutation. This property is straightforward from the definition of empirical probability measures. For any permutation function σ, we have $P_X = \frac{1}{m} \sum_{i=1}^{m} \delta_{x_i} = \frac{1}{m} \sum_{i=1}^{m} \delta_{x_{\sigma(i)}} = P_{\sigma(X)}$ which completes the proof. $\square$

We now continue the proof of Theorem 1. If $\mathbb{E}_{X \sim p(X)} \left(\mathbb{E}_{\theta \sim \text{vMF}(\epsilon,\kappa)} W_p^p(\theta\sharp P_X, \theta\sharp P_{g_\gamma(f_\phi(X))}) \right)^{\frac{1}{p}} = 0$, we obtain $\left(\mathbb{E}_{\theta \sim \text{vMF}(\epsilon,\kappa)} W_p^p(\theta\sharp P_X, \theta\sharp P_{g_\gamma(f_\phi(X))}) \right)^{\frac{1}{p}} = $ v-DSW$(X, g_\gamma(f_\phi(X)); \epsilon, \kappa) = 0$ for p-almost surely $X \in \mathcal{X}$. By Collorary 1, we obtain $X = g_\gamma(f_\phi(X))$ for p-almost surely $X \in \mathcal{X}$. We complete the proof.

B.3. Proof for Proposition 2

We first recall the definition of the self-attention amortized model in Definition 3:

$$a_\psi(X, Y) = \frac{\mathcal{A}_\zeta(X'^\top)^\top \mathbf{1}_m + \mathcal{A}_\zeta(Y'^\top)^\top \mathbf{1}_m}{||\mathcal{A}_\zeta(X'^\top)^\top \mathbf{1}_m + \mathcal{A}_\zeta(Y'^\top)^\top \mathbf{1}_m||_2},$$

Symmetry: Since the self-attention amortized model use the same attention weight ζ for both X and Y, exchanging X and Y yields the same results $a_\psi(X, Y) = a_\psi(Y, X)$.

Permutation invariance: Based on the results in Yang et al. (2019b, Appendix A), we show that self-attention amortized model is permutation invariant. In particular, we have:

$$\begin{aligned}
\mathcal{A}_\zeta(X'^\top)^\top \mathbf{1}_m &= \text{Att}(X'^\top W_q, X'^\top W_k, X'^\top W_v)^\top \mathbf{1}_m \\
&= \left(\text{softmax}_{\text{row}} \left[\frac{X'^\top W_q W_k^\top X'}{\sqrt{d_k}} \right] X'^\top W_v \right)^\top \mathbf{1}_m \\
&= \left(\text{softmax}_{\text{row}} \left[\frac{\sigma(X)'^\top W_q W_k^\top \sigma(X)'}{\sqrt{d_k}} \right] \sigma(X)'^\top W_v \right)^\top \mathbf{1}_m \\
&= \mathcal{A}_\zeta(\sigma(X)'^\top)^\top \mathbf{1}_m.
\end{aligned}$$

Similarly, the proof holds for both linear self-attention and efficient self-attention.

C. Experiment settings

In this section, we first provide the details of the training process and the architecture for point-cloud reconstruction, transfer learning, and point-cloud generation. Then, we present the implementation detail and hyper-parameters settings for different distances used in our experiments.

C.1. Details of point-cloud reconstruction and downstream applications

Point-cloud reconstruction: We use the same settings in ASW (Nguyen et al., 2021c) to train autoencoders. We utilize a variant of Point-Net (Qi et al., 2017) with an embedding size of 256 proposed in (Pham et al., 2020). The architecture of the autoencoder and classifier are shown in Figure 5. Our autoencoder is trained on the ShapeNet Core-55 dataset (Chang et al., 2015) with a batch size of 128 and a point-cloud size of 2048. We train it for 300 epochs using an SGD optimizer with an initial learning rate of 1e-3, a momentum of 0.9, and a weight decay of 5e-4.

Next, we detail the process of conducting two downstream applications of point-cloud reconstruction.

Transfer learning: A classifier is trained on the latent space of the autoencoder. Particularly, we extract a 256-dimension (which is smaller than the setting in (Lee et al., 2022)) latent vector of an input 3D point-cloud via the pre-trained encoder. Then, this vector is fed into a multi-layer perceptron with hidden layers of size 512 and 256. The last layer outputs a 40-dimension vector representing the prediction of 40 classes of the ModelNet40 dataset.
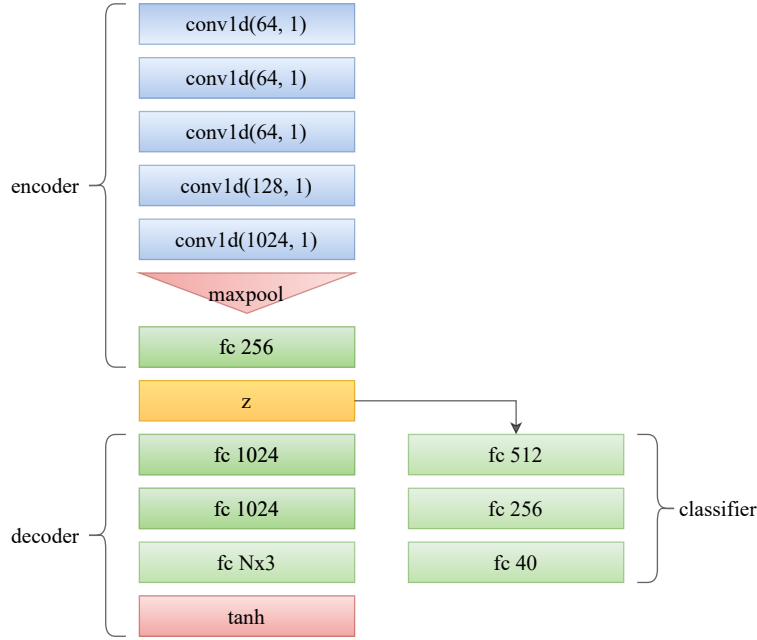
Figure 5. The architecture of the Point-Net variant in our experiments. For transfer learning, we use a simple classifier with 3 fully-connected layers. All layers are followed by ReLU activation and batch normalization by default, except for the final layers.

Point-cloud generation: Our generative model is trained on the latent space of the autoencoder as follows. First, we extract a 256-dimension latent vector of an input 3D point-cloud via the pre-trained encoder. Then a 64-dimensional vector is drawn from a normal distribution $\mathcal{N}(0, \mathbb{I}_{64})$, where $\mathbb{I}_{64}$ is the $64x64$ identity matrix, and fed into a generator which also outputs a 256-dimension vector. Finally, the generator learns by minimizing the optimal transport distance between the generated and ground truth latent codes.

C.2. Details of baseline distances

We want to emphasize that we use the same set of hyper-parameters reported in (Nguyen et al., 2021c) for Chamfer, EMD, SW, and Max-SW.

Chamfer and EMD: We use the CUDA implementation from (Yang et al., 2019a).

SW: We use the Monte Carlo estimation with 100 slices.

Max-SW: We use the projected sub-gradient ascent algorithm to optimize the projection. It is trained with an Adam optimizer with an initial learning rate of 1e-4. The number of iterations T is chosen from $\{1, 10, 50\}$.

Adaptive SW: We use Algorithm 1 in (Nguyen et al., 2021c) with the same set of parameters as follows: $N_0 = 2, s = 1, \epsilon = 0.5$, and $M = 500$.

v-DSW: We use stochastic projected gradient ascent algorithm to optimize the location vector ϵ in Equation 19 while we fix the concentration parameter κ to 1 for both v-DSW and all of its amortized versions. Similar to Max-SW, it is trained with an Adam optimizer with an initial learning rate of 1e-4. The number of iterations T is selected from $\{1, 10, 50\}$ based on the task performance. Intuitively, increasing the number of iterations leads to a better approximation that is closer to the optimal value but comes with an expensive computational cost. We also use the Monte Carlo estimation with 100 slices as in SW.

C.3. Details of amortized sliced Wasserstein distances

Linear, generalized linear, and non-linear models: We adopt the official implementations in (Nguyen & Ho, 2022a).

Self-attention-based models: We adapt the official implementations from their corresponding papers in our experiments. For all variants, d_v is set to 3, which equals the dimension of point-clouds while d_k is chosen from $\{16, 32, 64, 128\}$. In Equation 15, the projected dimension k is selected from $\{64, 128\}$.

Training amortized models: The learning rate is set to 1e-3 and the optimizer is set to Adam (Kingma & Ba, 2014) with

Table 4. Reconstruction and transfer learning performance of different autoencoders on the ModelNet40 dataset. For v-DSW and Max-SW, T denotes the number of projected sub-gradient ascent iterations. In Table 1, both v-DSW and Max-SW have T = 50 iterations. All reconstructed losses except EMD are multiplied by 100.

Method	CD $(10^{-2}, \downarrow)$	SW $(10^{-2}, \downarrow)$	EMD $(\downarrow)$	Acc $(\uparrow)$	Time $(\downarrow)$
CD	1.25 ± 0.03	681.20 ± 16.73	653.52 ± 10.43	86.28 ± 0.34	95
EMD	0.40 ± 0.00	94.54 ± 2.90	168.60 ± 1.57	88.45 ± 0.20	208
SW	0.68 ± 0.01	89.61 ± 3.88	191.12 ± 2.88	87.90 ± 0.27	106
Max-SW (T = 1)	0.69 ± 0.01	87.60 ± 0.95	190.88 ± 0.40	88.05 ± 0.23	97
Max-SW (T = 10)	0.69 ± 0.01	90.72 ± 0.58	192.82 ± 0.73	87.82 ± 0.37	102
Max-SW (T = 50)	0.68 ± 0.01	88.22 ± 1.45	190.23 ± 0.1	87.97 ± 0.14	116
ASW	0.69 ± 0.01	89.42 ± 5.07	192.03 ± 3.09	87.78 ± 0.20	103
v-DSW (T = 1)	$\mathbf{0.67 \pm 0.01}$	87.29 ± 1.49	188.52 ± 1.47	87.87 ± 0.28	115
v-DSW (T = 10)	0.68 ± 0.00	87.44 ± 1.07	189.97 ± 1.04	87.98 ± 0.23	205
v-DSW (T = 50)	$\mathbf{0.67 \pm 0.00}$	85.03 ± 3.31	187.75 ± 2.00	87.83 ± 0.40	633
$\mathcal{L}$-Max-SW	1.06 ± 0.03	121.85 ± 5.77	236.87 ± 3.42	87.70 ± 0.23	$\mathbf{94}$
$\mathcal{G}$-Max-SW	12.11 ± 0.29	851.07 ± 2.11	829.28 ± 5.53	87.49 ± 0.36	97
$\mathcal{N}$-Max-SW	7.38 ± 3.29	618.74 ± 153.87	648.32 ± 117.03	87.43 ± 0.15	96
$\mathcal{L}$v-DSW (ours)	0.68 ± 0.00	85.32 ± 0.54	188.32 ± 0.23	87.70 ± 0.34	114
$\mathcal{G}$v-DSW (ours)	0.68 ± 0.01	82.77 ± 0.48	187.04 ± 1.11	87.75 ± 0.19	117
$\mathcal{N}$v-DSW (ours)	$\mathbf{0.67 \pm 0.00}$	83.47 ± 0.49	186.66 ± 0.81	87.84 ± 0.07	115
$\mathcal{A}$v-DSW (ours)	0.67 ± 0.01	83.08 ± 1.22	186.27 ± 0.56	88.05 ± 0.17	230
$\mathcal{EA}$v-DSW (ours)	0.68 ± 0.01	82.05 ± 0.40	186.46 ± 0.25	88.07 ± 0.21	125
$\mathcal{LA}$v-DSW (ours)	0.68 ± 0.00	$\mathbf{81.03 \pm 0.18}$	$\mathbf{185.26 \pm 0.31}$	$\mathbf{88.28 \pm 0.13}$	123

Table 5. Quantitative results (measured in EMD) of reconstructing point-clouds in the ShapeNet Core-55 dataset.

Method	PC1	PC2	PC3	PC4	PC5	PC6	Avg
SW	141.07	139.50	118.83	99.11	150.28	128.46	129.54
Max-SW (T = 50)	145.15	131.76	112.13	116.73	139.91	115.79	126.91
ASW	139.17	126.55	115.49	$\mathbf{91.07}$	153.87	114.84	123.50
v-DSW (T = 50)	133.06	146.99	105.65	105.66	137.32	$\mathbf{110.50}$	123.20
$\mathcal{N}$v-DSW	132.60	127.57	100.81	94.31	131.04	116.34	117.11
$\mathcal{EA}$v-DSW	139.64	$\mathbf{124.28}$	100.34	98.33	$\mathbf{123.59}$	115.05	116.87
$\mathcal{LA}$v-DSW	$\mathbf{130.21}$	127.00	$\mathbf{96.75}$	98.09	132.33	114.11	$\mathbf{116.41}$

Table 6. Reconstruction results of SW and $\mathcal{LA}$v-DSW when changing L. CD and SWD are multiplied by 100.

Method	L	CD $(10^{-2}, \downarrow)$	SW $(10^{-2}, \downarrow)$	EMD $(\downarrow)$	Time
SW	50	$\mathbf{0.67 \pm 0.00}$	90.17 ± 2.97	190.97 ± 1.87	100
	100	0.68 ± 0.01	$\mathbf{89.61 \pm 3.88}$	191.12 ± 2.88	107
	200	$\mathbf{0.67 \pm 0.00}$	89.54 ± 4.57	191.21 ± 3.87	111
	500	0.67 ± 0.01	88.20 ± 4.22	190.14 ± 2.35	142
$\mathcal{LA}$v-DSW	50	0.68 ± 0.01	85.88 ± 4.03	188.80 ± 2.55	133
	100	0.68 ± 0.00	$\mathbf{81.03 \pm 0.18}$	$\mathbf{185.26 \pm 0.31}$	123

$(\beta_1, \beta_2) = (0, 0.9)$.

D. Additional experimental results

Point-cloud reconstruction: Table 4 illustrates the full quantitative results of the point-cloud reconstruction experiment. For Max-SW and v-DSW, we vary the number of gradient iterations T in $\{1, 10, 50\}$. Because CD is not a proper distance so we choose the best number of iterations based on SW and EMD losses (we prioritize EMD loss first then SW). As can be
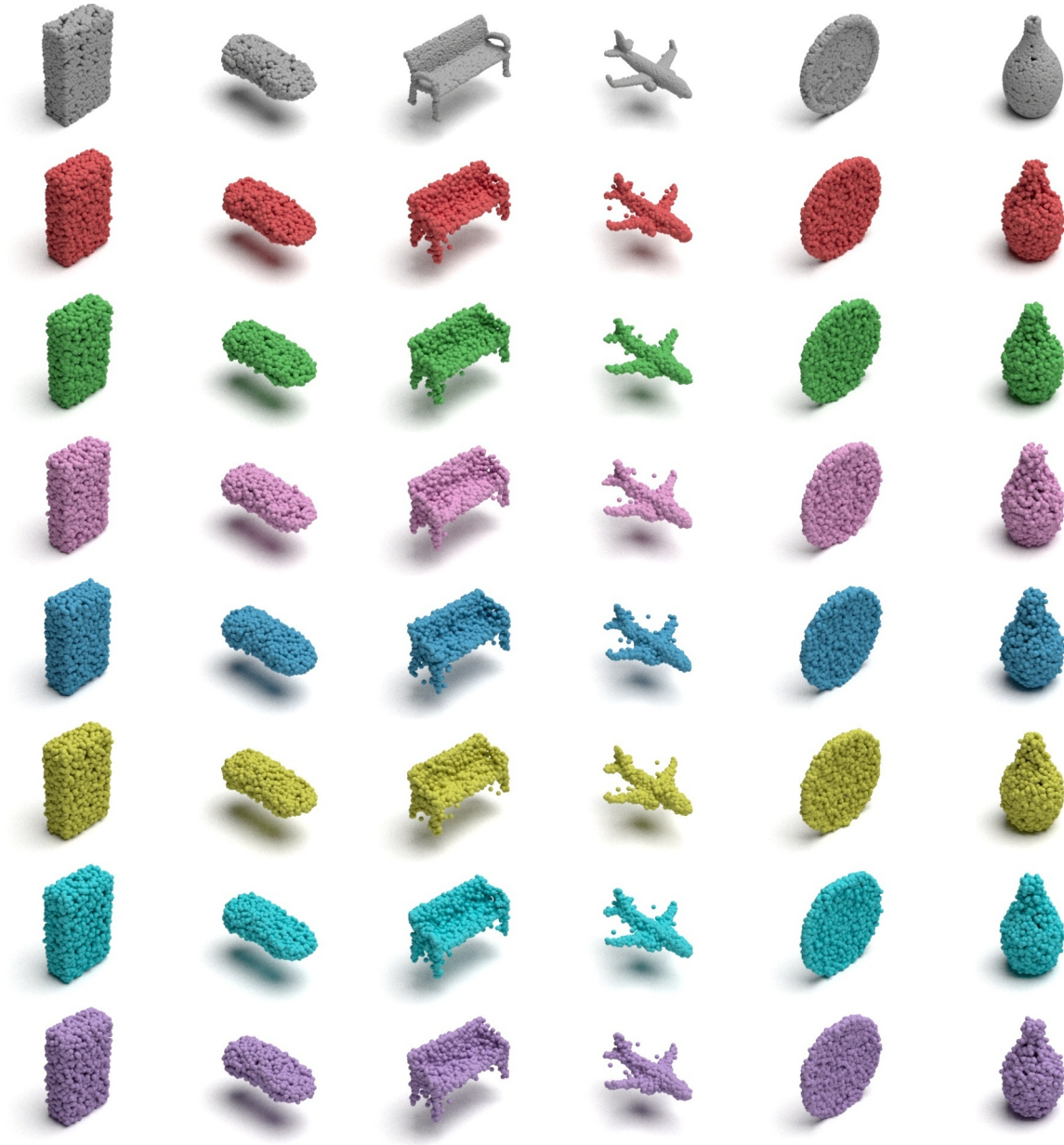
Figure 6. Qualitative results of reconstructing point-clouds in the ShapeNet Core-55 dataset. From top to bottom: input, SW, Max-SW (T = 50), ASW, v-DSW (T = 50), $\mathcal{N}$v-DSW, $\mathcal{E}\mathcal{A}$v-DSW, and $\mathcal{L}\mathcal{A}$v-DSW.

Table 7. Reconstruction results of v-DSW when changing the number of projected sub-gradient ascent iteration (T). CD and SWD are multiplied by 100.

Method	CD $(10^{-2},\downarrow)$	SW $(10^{-2},\downarrow)$	EMD $(\downarrow)$
v-DSW (T = 0)	0.67 ± 0.01	88.63 ± 2.30	189.81 ± 1.19
v-DSW (T = 1)	0.67 ± 0.01	87.29 ± 1.49	188.52 ± 1.47
v-DSW (T = 10)	0.68 ± 0.00	87.44 ± 1.07	189.97 ± 1.04
v-DSW (T = 50)	$\mathbf{0.67 \pm 0.00}$	85.03 ± 3.31	187.75 ± 2.00
$\mathcal{L}\mathcal{A}$v-DSW	0.68 ± 0.00	$\mathbf{81.03 \pm 0.18}$	$\mathbf{185.26 \pm 0.31}$

Table 8. Reconstruction results of $\mathcal{LA}$v-DSW when changing κ. CD and SWD are multiplied by 100.

κ	CD $(10^{-2}, \downarrow)$	SW $(10^{-2}, \downarrow)$	EMD $(\downarrow)$
0.1	**0.67 ± 0.00**	81.88 ± 1.09	185.30 ± 0.94
1	0.68 ± 0.00	**81.03 ± 0.18**	**185.26 ± 0.31**
10	0.85 ± 0.01	96.01 ± 4.24	208.46 ± 4.03

Table 9. Performance comparison of point-cloud generation on the chair category of ShapeNet. For v-DSW and Max-SW, T denotes the number of projected sub-gradient ascent iterations. In Table 3, v-DSW and Max-SW have T = 50 and 10 iterations, respectively. JSD, MMD-CD, and MMD-EMD are multiplied by 100.

Method	JSD $(10^{-2}, \downarrow)$	MMD $(10^{-2}, \downarrow)$		COV $(\%, \uparrow)$		1-NNA $(\%, \downarrow)$	
		CD	EMD	CD	EMD	CD	EMD
CD	17.88 ± 1.14	1.12 ± 0.02	17.19 ± 0.36	23.73 ± 1.69	10.83 ± 0.89	98.45 ± 0.10	100.00 ± 0.00
EMD	5.15 ± 1.52	0.61 ± 0.09	10.37 ± 0.61	41.65 ± 2.19	42.54 ± 2.42	87.76 ± 1.46	87.30 ± 1.22
SW	1.56 ± 0.06	0.72 ± 0.02	10.80 ± 0.11	38.55 ± 0.43	45.35 ± 0.48	89.91 ± 1.17	88.28 ± 0.70
Max-SW (T = 1)	1.74 ± 0.22	0.78 ± 0.05	11.05 ± 0.31	39.39 ± 2.28	46.82 ± 0.79	92.15 ± 0.95	90.20 ± 0.87
Max-SW (T = 10)	1.63 ± 0.32	0.74 ± 0.01	10.84 ± 0.08	40.47 ± 1.04	47.81 ± 0.78	91.46 ± 0.72	89.93 ± 0.86
Max-SW (T = 50)	1.57 ± 0.26	0.80 ± 0.05	11.25 ± 0.34	37.81 ± 1.69	46.23 ± 0.64	92.15 ± 0.72	90.35 ± 0.28
ASW	1.75 ± 0.38	0.78 ± 0.05	11.27 ± 0.38	38.16 ± 2.15	45.45 ± 1.40	91.21 ± 0.40	89.36 ± 0.40
v-DSW (T = 1)	1.84 ± 0.17	0.75 ± 0.03	11.02 ± 0.21	38.26 ± 1.46	45.35 ± 1.70	90.08 ± 0.48	87.81 ± 0.16
v-DSW (T = 10)	1.48 ± 0.17	0.77 ± 0.02	11.09 ± 0.09	37.22 ± 0.96	43.77 ± 0.39	90.40 ± 1.05	88.87 ± 1.04
v-DSW (T = 50)	1.79 ± 0.17	0.72 ± 0.02	10.73 ± 0.20	37.76 ± 0.71	45.49 ± 1.37	90.23 ± 0.13	88.33 ± 0.95
$\mathcal{L}$v-DSW (ours)	1.67 ± 0.07	0.77 ± 0.04	11.10 ± 0.33	37.91 ± 1.84	45.64 ± 2.30	90.42 ± 0.53	88.82 ± 0.38
$\mathcal{G}$v-DSW (ours)	1.56 ± 0.22	0.75 ± 0.02	10.99 ± 0.11	37.81 ± 1.70	45.69 ± 0.46	90.32 ± 0.38	88.26 ± 0.28
$\mathcal{N}$v-DSW (ours)	**1.44 ± 0.06**	0.75 ± 0.02	10.95 ± 0.09	38.40 ± 1.34	46.28 ± 2.06	90.15 ± 0.80	88.65 ± 0.82
$\mathcal{EA}$v-DSW (ours)	1.73 ± 0.21	**0.71 ± 0.04**	**10.70 ± 0.26**	40.03 ± 1.28	**48.01 ± 1.07**	89.98 ± 0.57	88.55 ± 0.38
$\mathcal{LA}$v-DSW (ours)	1.54 ± 0.09	0.72 ± 0.03	10.74 ± 0.35	**40.62 ± 1.39**	45.84 ± 1.23	**89.44 ± 0.28**	**87.79 ± 0.37**

seen from the table, increasing the number of gradient ascent iterations (T) increases the reconstruction performance of Max-SW and v-DSW but comes with the cost of additional computation, especially for v-DSW. However, with all choices of T, the reconstruction performance (measured in SW and EMD) of both Max-SW and v-DSW are generally worse than our amortized methods. In addition, our amortized methods have smaller standard deviations over 3 runs, thus they are more stable than the conventional optimization method using gradient ascent method. The qualitative results are given in Figure 6. The corresponding quantitative results in EMD are given in Table 5. It can be seen that our amortized v-DSW methods have more favorable performance.

On the number of projections (L). In our experiments, L is fixed to 100 as in the ASW's paper. Here, we conduct an ablation study on the number of projections L and report the result in Table 6. As can be seen from the table, increasing the number of projections improves the performance in terms of EMD but comes with an extra running time. We see that $\mathcal{LA}$v-DSW with $L = 50$ and $L = 100$ are faster than SW with $L = 500$ while being better in terms of SW and EMD evaluation metrics. Compared to SW with $L = 200$, $\mathcal{LA}$v-DSW with $L = 50$ has approximately the same computational time while having lower SW and EMD evaluation metrics.

On the choice of location vector ϵ. We would like to recall that the optimal location vector $\epsilon^\star$ of v-DSW are computed using Algorithm 4 in Appendix A.3. To show its effectiveness, we compare it with a random location ϵ, i.e. T = 0. Table 7 illustrates that optimizing for the location parameter of the vMF distribution helps to improve the reconstruction. Moreover, our amortized optimization still gives better reconstruction scores than the randomly initialized location ϵ and the optimized location using the conventional method. Therefore, using amortized optimization could actually have benefits.

On the choice of parameter κ. We would like first to recall that κ is set to 1 for all v-DSW and amortized v-DSW methods in our experiments. In practice, the parameter κ can be chosen by doing a grid search. Here, we conduct an ablation study by varying $\kappa \in \{0.1, 1, 10\}$ for $\mathcal{LA}$v-DSW and report the results in Table 8. As can be seen from the table, $\kappa = 1$ results in the best-performing EMD.

Point-cloud generation. We summarize the full quantitative results for point-cloud generation in Table 9. For Max-SW and v-DSW, we again change the number of gradient iterations T in $\{1, 10, 50\}$. Note that $\mathcal{A}$v-DSW cannot be used in this experiment due to being out of memory while the performance of amortized Max-SW is too bad. Therefore, their results are not reported in this experiment. As can be seen from the table, amortized v-DSW methods achieve the best performance in 7 out of 7 metrics. Using more than one gradient ascent iteration ($T \in \{10, 50\}$) does improve the generation performance of Max-SW and v-DSW but comes with the cost of additional computation.