

SLOSH: Set Locality Sensitive Hashing via Sliced-Wasserstein Embeddings

Yuzhe Lu*

Computer Science Department,
Vanderbilt University,
Nashville, TN, 37235

yuzhe.lu@vanderbilt.edu

Andrea Soltoggio

School of Computer Science,
Loughborough University,
Leicestershire, UK

a.soltoggio@lboro.ac.uk

Xinran Liu*

Computer Science Department,
Vanderbilt University,
Nashville, TN, 37235

xinran.liu@vanderbilt.edu

Soheil Kolouri

Computer Science Department,
Vanderbilt University,
Nashville, TN, 37235

soheil.kolouri@vanderbilt.edu

Abstract

Learning from set-structured data is an essential problem with many applications in machine learning and computer vision. This paper focuses on non-parametric and data-independent learning from set-structured data using approximate nearest neighbor (ANN) solutions, particularly locality-sensitive hashing. We consider the problem of set retrieval from an input set query. Such retrieval problem requires: 1) an efficient mechanism to calculate the distances/dissimilarities between sets, and 2) an appropriate data structure for fast nearest neighbor search. To that end, we propose Sliced-Wasserstein set embedding as a computationally efficient “set-2-vector” mechanism that enables downstream ANN, with theoretical guarantees. The set elements are treated as samples from an unknown underlying distribution, and the Sliced-Wasserstein distance is used to compare sets. We demonstrate the effectiveness of our algorithm, denoted as Set-LOcality Sensitive Hashing (SLOSH), on various set retrieval datasets and compare our proposed embedding with standard set embedding approaches, including Generalized Mean (GeM) embedding/pooling, Featurewise Sort Pooling (FSPool), and Covariance Pooling and show consistent improvement in retrieval results. The code for replicating our results is available here: <https://github.com/mint-vu/SLOSH>.

1. Introduction

The nearest neighbor search problem is at the heart of many nonparametric learning approaches in classification,

regression, and density estimation, with many applications in machine learning, computer vision, and other related fields [3, 6, 55]. The exhaustive search solution to the nearest neighbor problem for N given objects (e.g., images, vectors, etc.) requires N evaluation of (dis)similarities (or distances), which could be problematic when: 1) the number of objects, N , is large, or 2) (dis)similarity evaluation is expensive. Approximate Nearest Neighbor (ANN) [5] approaches have been proposed as an efficient alternative for similarity search on massive datasets. ANN approaches leverage data structures like random projections, e.g., Locality-Sensitive Hashing (LSH) [17, 21], or tree-based structures, e.g., kd-trees [10, 63], to reduce the complexity of nearest neighbor search. Ideally, ANN approaches must address both these challenges, i.e., decreasing the number of similarity evaluations and reducing the computational complexity of similarity calculations while providing theoretical guarantees on ANN retrievals.

Despite the great strides in developing ANN methods, the majority of the existing approaches are designed for objects living in Hilbert spaces. Recently, however, there has been an increasing interest in set-structured data with many applications in point cloud processing, graph learning, image/video recognition, and object detection, to name a few [36, 62, 70]. Even when the input data itself is not a set, in many applications, the complex input data (e.g., a natural image or a graph) is decomposed into a set of more abstract components (e.g., objects or node embeddings). Similarity search for large databases of set-structured data remains an active field of research, with many real-world applications. In this paper, we focus on developing a data-independent

ANN method for set-structured data. We leverage insights from **computational optimal transport** [13, 30, 48, 61] and propose a novel LSH algorithm, which relies on Sliced-Wasserstein Embeddings and enables efficient set retrieval.

Defining (dis)similarities for set-structured data comes with unique challenges: i) the sets could have different cardinalities, and ii) the set elements do not necessarily have an inherent ordering. Hence, a similarity measure for set-structured data must handle varied input sizes and should be invariant to permutations, i.e., the (dis)similarity score should not change under any permutation of the input set elements. Generally, the existing approaches for defining similarities between sets rely on the following two strategies. First, solving an assignment problem (via optimization) for finding corresponding elements between two sets and aggregate (dis)similarities between corresponding elements, e.g., using Hungarian algorithm, Wasserstein distances, Chamfer loss, etc. **These approaches are at best quadratic and at worst cubic in the set cardinalities.**

The second family of approaches rely on embedding the sets into a vector space and leveraging common similarities in the embedded space. The set embedding could be explicit (e.g., deep set networks) [36, 70] or implicit (e.g., Kernel methods) [11, 19, 22, 32, 42, 50, 51, 69]. Also, the embedding process could be data-dependent (i.e., learning based) as in deep set learning approaches, which leverage a composition of permutation-equivariant backbones followed by a permutation-invariant global pooling mechanisms that define a parametric permutation-invariant set embedding into a Hilbert space [36, 70, 72]. Or, it can be data-independent as it is the case for global average/max/sum/covariance pooling, variations of Janossy pooling [43], and variations of Wasserstein embedding [27], among others. Recently, there has been a lot of interest in learning-based embeddings using deep neural networks and in particular transformer networks. However, data-independent embedding approaches (e.g., global poolings) have received less attention.

Contributions. Our paper **focuses on non-parametric learning from set-structured data using data-independent set embeddings.** Precisely, we consider the problem where our training data is a **set of sets**, i.e., $\mathcal{X} = \{X_i | X_i = \{x_n^i \in \mathbb{R}^d\}_{n=0}^{N_i-1}\}_{i=1}^I$, (e.g., set of point clouds), and **for a query set X we would like to retrieve the K -Nearest Neighbors (KNN) from \mathcal{X} .** To solve this problem, we require a fast and reliable (dis)similarity measure between sets, and a computationally efficient nearest neighbor search. We propose **Sliced-Wasserstein Embedding as a computationally efficient and powerful tool that provides a set-2-vector operator** with computational complexity of $\mathcal{O}(LN(d + \log(N)))$, with sequential processing, $\mathcal{O}(N(d + \log(N)))$, with parallel processing, where **L is of the same order as d .** Treating sets as empirical distributions, **Sliced-Wasserstein Embedding embeds sets in a vector space in which the Eu-**

clidean distance between two embedded vectors is equal to the Sliced-Wasserstein distance between their corresponding empirical distributions. Such embedding enables the application of fast ANN approaches, like Locality Sensitive Hashing (LSH), to sets while providing collision probabilities with respect to the Sliced-Wasserstein distance. Finally, we provide extensive numerical results analyzing and comparing our approach with various data-independent embedding methods in the literature.

2. Related Work

Set embeddings (set-2-vector): Machine learning on set-structured data is challenging due to: 1) permutation-invariant nature of sets, and 2) having various cardinalities. Hence, any model (parametric or non-parametric) designed for analyzing set-structured data has to be permutation invariant, and allow for inputs of various sizes. Today, a common approach for learning from sets is to use a permutation equivariant parametric function, e.g., fully connected networks [70] or transformer networks [36], composed with a permutation invariant function, i.e., a global pooling, e.g., global average pooling, or pooling by multi-head attention [36]. One can view this process as embedding a set into a fixed-dimensional representation through a parametric embedding that could be learned using the training data and an objective function (e.g., classification).

A major unanswered question is regarding non-parametric learning from set-structured data. In other words, what would be a good data-independent set embedding that one can use for generic applications, including K-Nearest Neighbor classification/regression/density estimation? Given that a data-independent global pooling could be viewed as a set-2-vector process, we surveyed the existing set-2-vector mechanisms in the literature. In particular, global average/max/sum and covariance pooling [64] could be considered as the simplest such processes. Generalized Mean (GeM) [53] is another pooling mechanism commonly used in image retrieval applications, which captures higher statistical moments of the underlying distributions. Other notable approaches include VLAD [7, 23], CroW [25], and FSPool [72], among others.

Locality Sensitive Hashing (LSH): A LSH function hashes two “similar” objects into the same bucket with “high” probability, while ensuring that “dissimilar” objects will end up in the same bucket with “low” probability. Originally presented in [21] and extended in [17], LSH uses random projections of high-dimensional data to hash samples into different buckets. The LSH algorithm forms the foundation of many ANN search methods, which provide theoretical guarantees and have been extensively studied since its conception [3, 4, 6, 33].

Here, we are interested in nearest neighbor retrieval for sets. More precisely, given a training *set* of sets as training

L: 也许是进行投影的次数?

data (think of *set* of point clouds), and a test set (a point cloud representation of an object) we would like to retrieve the “nearest” sets in our training *set* in an efficient manner. To that end, we extend LSH to enable its application to set retrieval. While there has been a few recent work [26, 45] on the topic of LSH for set queries, our proposed approach significantly differs from these work. In contrast to [26, 45], we provide a Euclidean embedding for sets, which allows for a direct utilization of the LSH algorithm and provides collision probabilities as a function of the set metrics.

Wasserstein-based learning: Wasserstein distances are rooted in the **optimal transportation problem** [30, 48, 61], and they provide a robust mathematical framework for comparing probability distributions that respect the underlying geometry of the space. Wasserstein distances have recently received abundant interest from the Machine Learning and Computer Vision communities. These distances and their variations (e.g., Sliced-Wasserstein distances [13, 52] and subspace robust Wasserstein distances [47]) have been extensively studied in the context of deep generative modeling [8, 20, 31, 38, 60], domain adaptation [9, 15, 16, 35], transfer learning [2], adversarial attacks [66, 67], and adversarial robustness [37, 58].

More recently, Wasserstein distances and optimal transport have been used in the context of comparing set-structured data. The main idea behind these recent approaches is **to treat sets (with possibly variable cardinalities) as empirical distributions and use transport-based distances for comparing/modeling these distributions**. For instance, Togninalli et al. [59] propose to compare node embeddings of two graphs (treated as sets) via the Wasserstein distance. Later, Mialon et al. [40] and Kolouri et al. [27] concurrently propose Wasserstein embedding frameworks for extracting fixed-dimensional representations from set-structured data. Here, we further extend this direction and propose Sliced-Wasserstein Embedding as a computationally efficient approach that allows us to perform data-independent non-parametric learning from set-structured data.

$p_i(x)$: 表现了 x_i 的分布, x_i 的密度 (density) 函数, μ_i : 表示distribution

3. Preliminaries

We denote an input set with N_i elements living in \mathbb{R}^d by $X_i = \{x_n^i \in \mathbb{R}^d\}_{n=0}^{N_i-1}$. We view sets as empirical probability measures, μ_i , defined in $\mathbb{X} \subseteq \mathbb{R}^d$ with probability density $d\mu_i(x) = p_i(x)dx$, where $p_i(x) = \frac{1}{N_i} \sum_{n=1}^{N_i} \delta(x - x_n^i)$, and $\delta(\cdot)$ is the Dirac delta function. The main idea is then to define the distance between two sets, X_i and X_j , as a probability metric between their empirical distributions.

3.1. Sliced-Wasserstein distances

We use the Sliced-Wasserstein (SW) probability metric as a distance measure between the sets (viewed as empirical probability distributions). Later we will see that **this choice allows us to effectively embed sets into a vector space such**

that the Euclidean distance between embedded sets is equal to the SW-distance between their corresponding empirical distributions. But first, let us briefly define the Wasserstein and Sliced-Wasserstein distances.

Let μ_i and μ_j be one-dimensional probability measures defined on \mathbb{R} . Then the p -Wasserstein distance between these measures can be written as:

$$\mathcal{W}_p(\mu_i, \mu_j) = \left(\int_0^1 (F_{\mu_i}^{-1}(\tau) - F_{\mu_j}^{-1}(\tau))^p d\tau \right)^{\frac{1}{p}} \quad (1)$$

where F_{μ}^{-1} is the inverse of the cumulative distribution function (c.d.f) F_{μ} of μ , i.e., it is the **quantile function**. The **one-dimensional p -Wasserstein distance for empirical distributions with N and M samples** can be computed with $\mathcal{O}(N \log(N) + M \log(M))$, which is in stark difference from the generally cubic order for ($d > 1$)-dimensional distributions. The one-dimensional case motivates the concept of Sliced-Wasserstein distances [28, 52]. For the rest of this paper we will consider only the case $p = 2$, **and for brevity we refer to 2-Wasserstein and 2-Sliced-Wasserstein distances as Wasserstein and Sliced-Wasserstein distances.**

The main idea behind SW distances is to slice d -dimensional distributions into infinite sets of their one-dimensional slices/marginals and then calculate the expected Wasserstein distance between their slices. **Let $g_{\theta} : \mathbb{R}^d \rightarrow \mathbb{R}$ be a parametric function with parameters $\theta \in \Omega_{\theta} \subseteq \mathbb{R}^{d_{\theta}}$, satisfying the regularity conditions in both inputs and parameters as presented in [28]. A common choice is $g_{\theta}(x) = \theta^T x$ where $\theta \in \mathbb{S}^{d-1}$ is a unit vector in \mathbb{R}^d , and \mathbb{S}^{d-1} denotes the unit d -dimensional hypersphere.** The slice of a probability measure, μ , with respect to g_{θ} is the one-dimensional probability measure $g_{\theta\#}\mu$, with the density,

$$p_i^{\theta}(t) := \int_{\mathbb{X}} \delta(t - g_{\theta}(x)) d\mu_i(x) \quad \forall t \in \mathbb{R}. \quad (2)$$

The generalized Sliced-Wasserstein distance is defined as

$$g_{SW_2}(\mu_i, \mu_j) = \left(\int_{\Omega_{\theta}} \mathcal{W}_2^2(g_{\theta\#}\mu_i, g_{\theta\#}\mu_j) d\sigma(\theta) \right)^{\frac{1}{2}}, \quad (3)$$

where $\sigma(\theta)$ is the uniform measure on Ω_{θ} , and once again for $g_{\theta}(x) = \theta^T x$ and $\Omega_{\theta} = \mathbb{S}^{d-1}$, the generalized Sliced-Wasserstein distance is simply the Sliced-Wasserstein distance. Equation (3) is the expected value of the Wasserstein distances between uniformly distributed slices (i.e., on Ω_{θ}) of distributions μ_i and μ_j .

3.2. Locality Sensitive Hashing

Locality Sensitive Hashing (LSH) [17, 21] strives for hashing near points in the high-dimensional input-space into the same hash bucket with a higher probability than distant ones, effectively solving the (R, c) -Near Neighbor problem. More precisely, let $H := \{h : \mathbb{R}^d \rightarrow U\}$ denote a

常规的投影: 线性投影

投影结果在参数空间上积分, 通常采用蒙特卡洛方法

LSH的性质

LSH function family with U denoting the hash values. The LSH function family H , is called (R, c, P_1, P_2) -sensitive if for any two points $u, v \in \mathbb{R}^d$ and $\forall h \in H$ it satisfies the following conditions:

- If $\|u - v\|_2 \leq R$, then $\Pr[h(u) = h(v)] \geq P_1$, and
- If $\|u - v\|_2 > cR$, then $\Pr[h(u) = h(v)] \leq P_2$

Where for a proper LSH $c > 1$ and $P_1 > P_2$. The original LSH for Euclidean distance uses,

$$h_{a,b}(u) = \lfloor \frac{a^T u + b}{\omega} \rfloor$$

where $\lfloor \cdot \rfloor$ is the floor function, $a \in \mathbb{R}^d$ is a random vector generated from a p -stable distribution (e.g., Normal or Cauchy distributions) [17], and b is a real number chosen uniformly from $[0, \omega)$ such that ω is the width of the hash bucket. Let $r = \|u - v\|_p$ and $f_p(t)$ denote the probability density function of the absolute value of the p -stable distribution, then this family of hash functions leads to the following probability of collision:

$$\Pr[h_{a,b}(u) = h_{a,b}(v)] = \int_0^\omega \frac{1}{r} f_p\left(\frac{t}{r}\right) \left(1 - \frac{t}{\omega}\right) dt \quad (4)$$

To amplify the gap between P_1 and P_2 one can concatenate several such hash functions. Let $\mathcal{G} := \{g : \mathbb{R}^d \rightarrow U^k\}$ denote the concatenation of k randomly selected hash functions, i.e., $g(u) = (h_1(u), \dots, h_k(u))$. In practice, multiple such hash functions g_1, \dots, g_T are often used. For points, u and v , within R -distance from one another, the probability that they collide at least in one of g_j s is: $1 - (1 - P_1^k)^T$.

Another commonly used, and related, family of hash functions consist of random projections and thresholding, i.e., $h_{a,b}(u) = \text{sgn}(a^T u + b)$ [14]. Using k such random projections (i.e., binary codes of length k) provides the following collision probability:

$$\Pr[g(u) = g(v)] = \left[1 - \frac{\cos^{-1}(u^T v)}{\pi}\right]^k$$

Since the conception of its idea [17, 21], many variants of LSH have been proposed. However, these approaches are not designed to handle set queries. Here, we extend LSH to set-structured data via Sliced-Wasserstein Embeddings.

4. Problem Formulation and Method

4.1. Sliced-Wasserstein Embedding

The idea of Sliced-Wasserstein Embeddings (SWE) is rooted in Linear Optimal Transport [30, 41, 65] and was first introduced in the context of pattern recognition from 2D probability density functions (e.g., images) [29] and more recently in [56]. Our work extends the framework to d -dimensional distributions with the specific application of set

retrieval in mind. Consider a set of probability measures $\{\mu_i\}_{i=1}^I$ with densities $\{p_i\}_{i=1}^I$, and recall that we use μ_i to represent the i 'th set $X_i = \{x_n^i\}_{n=0}^{N_i-1}$. At a high level, SWE can be thought as a set-2-vector operator, ϕ , such that:

$$\|\phi(\mu_i) - \phi(\mu_j)\|_2 = \mathcal{GSW}_2(\mu_i, \mu_j). \quad (5)$$

generalized

For convenience, we use $\mu_i^\theta := g_\theta \# \mu_i$ to denote the slice of measure μ_i with respect to g_θ . Also, let μ_0 denote a reference measure, with μ_0^θ being its corresponding slice. The optimal transport coupling (i.e., Monge coupling) between μ_i^θ and μ_0^θ can be written as

$$\text{transport plan: minimize} \quad T_i^\theta = F_{\mu_i^\theta}^{-1} \circ F_{\mu_0^\theta}, \quad (6)$$

and recall that $F_{\mu_i^\theta}^{-1}$ is the quantile function of μ_i^θ . Now, letting id denote the identity function, we define the so-called cumulative distribution transform (CDT) [46] of μ_i^θ as

$$\phi_\theta(\mu_i) := (T_i^\theta - id). \quad (7)$$

For a fixed θ , we can show that $\phi^\theta(\mu_i)$ satisfies (see supplementary material):

C1. The weighted ℓ_2 -norm of the embedded slice, $\phi_\theta(\mu_i)$, satisfies:

$$\begin{aligned} \|\phi_\theta(\mu_i)\|_{\mu_0^\theta, 2} &= \left(\int_{\mathbb{R}} \|\phi_\theta(\mu_i(t))\|_2^2 d\mu_0^\theta(t) \right)^{\frac{1}{2}} \\ &= \mathcal{W}_2(\mu_i^\theta, \mu_0^\theta), \end{aligned}$$

As a corollary we have $\|\phi_\theta(\mu_0)\|_{\mu_0^\theta, 2} = 0$.

C2. The weighted ℓ_2 distance between two embedded slices satisfies:

$$\|\phi_\theta(\mu_i) - \phi_\theta(\mu_j)\|_{\mu_0^\theta, 2} = \mathcal{W}_2(\mu_i^\theta, \mu_j^\theta). \quad (8)$$

It follows from **C1** and **C2** that:

$$\mathcal{GSW}_2(\mu_i, \mu_j) = \left(\int_{\Omega_\theta} \|\phi_\theta(\mu_i) - \phi_\theta(\mu_j)\|_{\mu_0^\theta, 2}^2 d\sigma(\theta) \right)^{\frac{1}{2}} \quad (9)$$

For probability measure μ_i , we then define the mapping to the embedding space via,

$$\phi(\mu_i) := \{\phi_\theta(\mu_i) \mid \theta \in \Omega_\theta\}.$$

Finally, we can re-weight the embedding (according to $d\mu_0^\theta$) such that the weighted ℓ_2 in 8 becomes the ℓ_2 distance as in 5. Next, we describe this reweighting and other implementation considerations in more details.

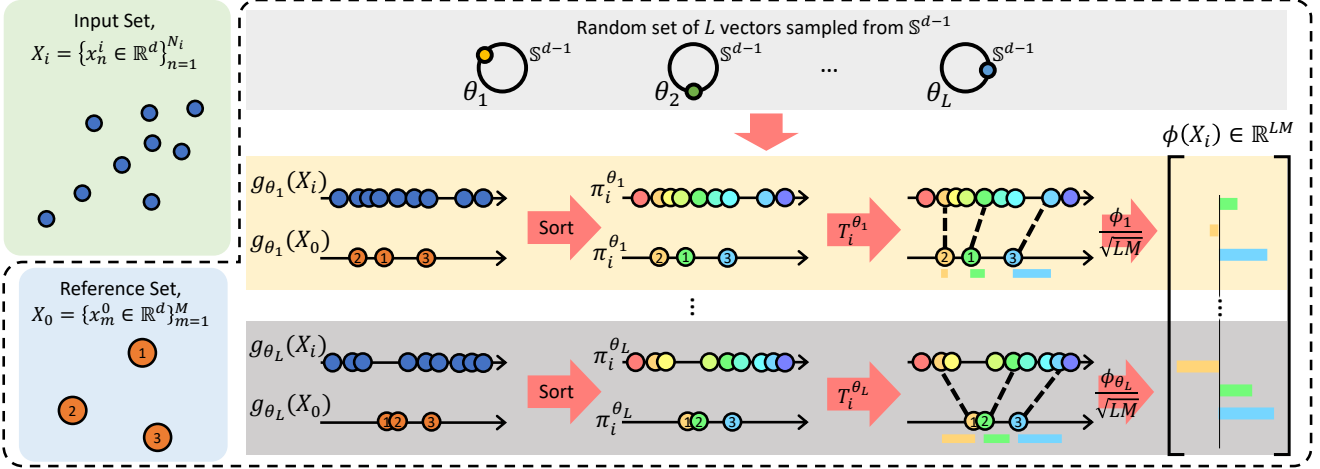


Figure 1. A graphical depiction of Sliced-Wasserstein Embedding (SWE) for a given reference set and a chosen number of slices, L . The input and reference sets are sliced via L random projections $\{g_{\theta_l}\}_{l=1}^L$. The projections are then sorted and the Monge couplings between input and reference sets' slices are calculated following Eq. (13). Finally, the embedding is obtained by weighting and concatenating ϕ_{θ_l} s. **The Euclidean distance between two embedded sets is equal to their corresponding $\widehat{\mathcal{GSW}}_{2,L}$ distance/dissimilarity measure, i.e., $\|\phi(X_i) - \phi(X_j)\|_2 = \widehat{\mathcal{GSW}}_{2,L}(X_i, X_j) \approx \mathcal{GSW}_2(X_i, X_j)$.**

Sliced Wasserstein Embedding

4.2. Monte Carlo Integral Approximation

The GSW distance in Eq. (3) relies on integration on Ω_θ (e.g., \mathbb{S}^{d-1} for linear slices), which cannot be directly calculated. Following the common practice in the literature [12, 31, 32, 39, 52], here, **we approximate the integration on θ via a Monte-Carlo (MC) sampling of Ω_θ** . Let $\Theta_L = \{\theta_l \sim \sigma(\theta)\}_{l=1}^L$ denote a set of L parameters sampled independently and uniformly from Ω_θ . We assume an empirical reference measure, $\mu_0 = \frac{1}{M} \sum_{m=1}^M \delta(x - x_m^0)$ with M samples. The MC approximation is written as:

$$\widehat{\mathcal{GSW}}_{2,L}^2(\mu_i, \mu_j) = \frac{1}{LM} \sum_{l=1}^L \|\phi_{\theta_l}(\mu_i) - \phi_{\theta_l}(\mu_j)\|_2^2. \quad (10)$$

Finally, our SWE embedding is calculated via:

$$\phi(\mu_i) = \left[\frac{\phi_{\theta_1}(\mu_i)}{\sqrt{LM}}; \dots; \frac{\phi_{\theta_L}(\mu_i)}{\sqrt{LM}} \right] \in \mathbb{R}^{LM \times 1}, \quad (11)$$

which satisfies:

$$\|\phi(\mu_i) - \phi(\mu_j)\|_2 = \widehat{\mathcal{GSW}}_{2,L}(\mu_i, \mu_j) \approx \mathcal{GSW}_2(\mu_i, \mu_j).$$

As for the approximation error, we rely on Theorem 6 in [44], which **uses Hölder's inequality and the results on the moments of the Monte Carlo estimation error to obtain:**

$$\mathbb{E}[\|\widehat{\mathcal{GSW}}_{2,L}^2(\mu_i, \mu_j) - \mathcal{GSW}_2^2(\mu_i, \mu_j)\|] \leq \sqrt{\frac{\text{var}(\mathcal{W}_2^2(\mu_i^\theta, \mu_j^\theta))}{L}} \quad (12)$$

The upper bound indicates that the approximation error decreases with \sqrt{L} . **The numerator, however, is implicitly dependent on the dimensionality of input space. Meaning that a larger number of slices, L , is needed for higher dimensions.**

4.3. SWE Algorithm

Here we review the algorithmic steps to obtain SWE. We consider $X_i = \{x_n^i\}_{n=0}^{N_i-1}$ as the **input set with N_i elements**, and $X_0 = \{x_m^0\}_{m=0}^{M-1}$ denote the **reference set of M samples** where in general $M \neq N_i$. For a fixed slicer g_θ we calculate $\{g_\theta(x_n^i)\}_{n=0}^{N_i-1}$ and $\{g_\theta(x_m^0)\}_{m=0}^{M-1}$ and **sort them increasingly**. Let π_i and π_0 denote the permutation indices (obtained from argsort). **Also, let π_0^{-1} denote the ordering that permutes the sorted set back to the original ordering**. Then we numerically calculate the **Monge coupling T_i^θ** via:

$$T_i^\theta[m] = F_{\mu_i^\theta}^{-1} \left(\frac{\pi_0^{-1}[m] + 1}{M} \right) \quad (13)$$

where $F_{\mu_0^\theta}(x_m^0) = \frac{\pi_0^{-1}[m] + 1}{M}$, assuming that the indices start from 0. Here $F_{\mu_i^\theta}^{-1}$ is calculated via interpolation. In our experiments we used linear interpolation similar to [38]. Note that **the dimensionality of the Monge coupling is only a function of the reference cardinality**, i.e., $T_i^\theta \in \mathbb{R}^M$. Consequently, we write:

$$\phi_\theta(X_i)[m] = (T_i^\theta[m] - g_\theta(x_m^0)) \quad (14)$$

and repeat this process for $\theta \in \{\theta_l \sim \sigma(\theta)\}_{l=1}^L$, while we emphasize that this process can be parallelized. The final embedding is achieved via weighting and concatenating ϕ_{θ_l} s as in Eq. (11), where the coefficient $\frac{1}{\sqrt{LM}}$ allows us to simplify the weighted Euclidean distance, $\|\cdot\|_{\mu_0,2}$, to Euclidean distance, $\|\cdot\|_2$. Algorithm 1 summarizes the embedding process, and Figure 1 provides a graphical depiction of

Algorithm 1 Sliced-Wasserstein Embedding

```

procedure SWE( $X_i = \{x_n^i\}_{n=0}^{N_i-1}$ ,  $X_0 = \{x_m^0\}_{m=0}^{M-1}$ ,  $L$ )
  Generate a set of  $L$  samples  $\Theta_L = \{\theta_l \sim \mathcal{U}_{\Omega_\theta}\}_{l=1}^L$ 
  Calculate  $g_{\Theta_L}(X_0) := \{g_{\theta_l}(x_m^0)\}_{m,l} \in \mathbb{R}^{M \times L}$ 
  Calculate  $\pi_0 = \text{argsort}(g_{\Theta_L}(X_0))$  and  $\pi_0^{-1}$  (on  $m$ -axis)
  Calculate  $g_{\Theta_L}(X_i) := \{g_{\theta_l}(x_n^i)\}_{n,l} \in \mathbb{R}^{N_i \times L}$ 
  Calculate  $\pi_i = \text{argsort}(g_{\Theta_L}(X_i))$  (on  $n$ -axis)
  for  $l = 1$  to  $L$  do
    Calculate the Monge coupling  $T_i^{\theta_l} \in \mathbb{R}^M$  (Eq. (13))
  end for
  Calculate the embedding  $\phi(X_i) \in \mathbb{R}^{M \times L}$  (Eq. (11))
  return  $\phi(X_i)$ 
end procedure

```

the process. Lastly, the SWE’s computational complexity for a set with cardinality $|X| = N$ is $\mathcal{O}(LN(d + \log N))$, where we assumed the cardinality of the reference set is of the same order as N . Note that $\mathcal{O}(LNd)$ is the cost of slicing and $\mathcal{O}(LN \log(N))$ is the sorting and interpolation cost to calculate Eq. (13).

4.4. SLOSH

Our proposed Set LOcality Sensitive Hashing (SLOSH) leverages the Sliced-Wasserstein Embedding (SWE) to embed training sets, $\mathcal{X}_{train} = \{X_i | X_i = \{x_n^i \in \mathbb{R}^d\}_{n=0}^{N_i-1}\}$, into a vector space where we can use Locality Sensitive Hashing (LSH) to perform ANN search. We treat each input set X_i as a probability measure $\mu_i(x) = \frac{1}{N_i} \sum_{n=1}^{N_i} \delta(x - x_n^i)$. For a reference set X_0 with cardinality $|X_0| = M$ and L slices, we embed the input set X_i into a (\mathbb{R}^{LM}) -dimensional vector space using Algorithm 1. With abuse of notation we use $\phi(\mu_i)$ and $\phi(X_i)$ interchangeably.

Given SWE, a family of (R, c, P_1, P_2) -sensitive LSH functions, H , will induce the following conditions,

- If $\widehat{\mathcal{GSW}}_{2,L}(X_i, X_j) \leq R$, then $\Pr[h(\phi(X_i)) = h(\phi(X_j))] \geq P_1$, and
- If $\widehat{\mathcal{GSW}}_{2,L}(X_i, X_j) > cR$, then $\Pr[h(\phi(X_i)) = h(\phi(X_j))] \leq P_2$

As mentioned in Section 2, for amplifying the gap between P_1 and P_2 , one uses $g(X_i) = [h_1(\phi(X_i)), \dots, h_k(\phi(X_i))]$, which results in a code length, k , for each input set, X_i . Finally, if $\widehat{\mathcal{GSW}}_{2,L}(X_i, X_j) \leq R$, by using T such codes, g_t for $t \in \{1, \dots, T\}$, of length k , we can ensure collision at least in one of g_t s with probability $1 - (1 - P_1^k)^T$.

5. Experiments

We evaluated our set locality sensitive hashing via Sliced-Wasserstein Embedding algorithm against using Generalized Mean (GeM) pooling [53], Global Covariance (Cov) pooling [64], and Featurewise Sort Pooling (FSPool) [71] on various set-structured datasets. We note that while FSPool was proposed as a data-dependent embedding, here we devise its data-independent variation for fair comparison. Interestingly, FSPool can be thought as a special case of our SWE embedding where $L = d$ and Θ_L is chosen as the identity matrix. To evaluate these methods, we tested all approaches on point cloud MNIST dataset (2D) [18, 34], ModelNet40 dataset (3D) [68], and the Oxford Buildings dataset [49] (i.e., Oxford 5k).

5.1. Baselines

Let $X_i = \{x_n^i \in \mathbb{R}^d\}_{n=0}^{N_i-1}$ be the input set with N_i elements. We denote $[X_i]_k = \{[x_n^i]_k\}_{n=0}^{N_i-1}$ as the set of all elements along the k ’th dimension, $k \in \{1, 2, \dots, d\}$. Below we provide a quick overview of the baseline approaches, which provide different set-2-vector mechanisms.

Generalized-Mean Pooling (GeM) [53] was originally proposed as a generalization of global mean and global max pooling on Convolutional Neural Network (CNN) features to boost image retrieval performance. Given the input X_i , GeM calculates the (p -th)-moment of each feature, $f^{(p)} \in \mathbb{R}^d$, as:

$$[f^{(p)}]_k = \left(\frac{1}{N_i} \sum_{n=1}^{N_i} ([x_n^i]_k)^p \right)^{\frac{1}{p}} \quad (15)$$

When pooling parameter $p = 1$, we end up with average pooling. While as $p \rightarrow \infty$, we get max pooling. In practice, we found that a concatenation of higher-order GeM features, i.e., $\phi_{GeM}(X_i) = [f^{(1)}; \dots; f^{(p)}] \in \mathbb{R}^{pd}$, leads to the best performance, where p is GeM’s hyper-parameter.

Covariance Pooling [1, 64] presents another way to capture second-order statistics and provide more informative representations. It was shown that this mechanism can be applied to CNN features as an alternative to global mean/max pooling to generate state-of-the-art results on facial expression recognition tasks [1]. Given input set X_i , the unbiased covariance matrix is computed by:

$$C = \frac{1}{N_i - 1} \sum_{n=1}^{N_i} (x_n^i - \bar{\mu}_i)(x_n^i - \bar{\mu}_i)^T, \quad (16)$$

where $\bar{\mu}_i = \frac{1}{N_i} \sum_{n=1}^{N_i} x_n^i$. The output matrix can be further regularized by adding a multiple of the trace to diagonal entries of the covariance matrix to ensure symmetric positive definiteness (SPD), $C_\lambda = C + \lambda \text{trace}(C) \mathbf{I}$ where λ is a regularization hyper-parameter and \mathbf{I} is the identity matrix. Covariance pooling then uses $\phi_{Cov}(X_i) = \text{flatten}(C_\lambda)$.

Table 1. Retrieval results of baselines and our approach on set retrieval on the three data sets.

	Point MNIST 2D			ModelNet 40			Oxford 5k		
	Precision@k/Accuracy			Precision@k/Accuracy			Precision@k/Accuracy		
	k=4	k=8	k=16	k=4	k=8	k=16	k=4	k=8	k=16
GeM-1 (GAP)	0.10/0.11	0.10/0.10	0.10/0.10	0.16/0.19	0.16/0.22	0.16/0.25	0.29/0.35	0.25/0.31	0.22/0.29
GeM-2	0.29/0.32	0.28/0.35	0.29/0.37	0.29/0.34	0.26/0.34	0.23/0.33	0.38/0.53	0.31/0.40	0.27/0.38
GeM-4	0.39/0.45	0.39/0.47	0.38/0.49	0.31/0.37	0.28/0.38	0.25/0.37	0.35/0.41	0.33/0.51	0.29/0.42
Cov	0.25/0.27	0.25/0.28	0.25/0.28	0.37/0.42	0.35/0.44	0.32/0.44	0.35/0.55	0.30/0.37	0.26/0.33
FSPool	0.75/0.80	0.74/0.81	0.73/0.81	0.50/0.57	0.47/0.58	0.42/0.56	0.47/0.53	0.39/0.6	0.32/0.49
SLOSH ($L < d$)	0.52/0.59	0.51/0.61	0.49/0.61	0.22/0.25	0.20/0.27	0.19/0.27	0.33/0.42	0.30/0.49	0.24/0.36
SLOSH ($L = d$)	0.67/0.73	0.64/0.74	0.62/0.73	0.39/0.44	0.36/0.45	0.33/0.44	0.43/0.58	0.39/0.60	0.34/0.55
SLOSH ($L > d$)	0.90/0.92	0.88/0.92	0.87/0.91	0.55/0.61	0.51/0.60	0.46/0.57	0.53/0.69	0.45/0.65	0.38/0.64

Featurewise Sort Pooling (FSPool) [71] is a powerful technique for learning representations from set-structured data. In short, this approach is based on sorting features along all elements of a set, $[X_i]_k$:

$$f = [\text{Sorted}([X_i]_1), \dots, \text{Sorted}([X_i]_d)] \in \mathbb{R}^{N_i \times d} \quad (17)$$

The fixed-dimensional representation is then obtained via an interpolation along the N_i dimension of f . More precisely, a continuous linear operator W is learned and the inner product between this continuous operator and f is evaluated at M fixed points (i.e., leading to weighted summation over d), resulting in a M -dimensional embedding.

Given that we are interested in a data-independent set representation, we cannot rely on learning the continuous linear operator W . Instead, we perform interpolation along the N_i axis on M points and drop the inner product all together to obtain a $(M \times d)$ -dimensional data-independent set representation. The mentioned variation of FSPool is similar to the Sliced-Wasserstein Embedding when $L = d$ and $\Theta_L = I_{d \times d}$, i.e., axis-aligned projections.

5.2. Datasets

Point Cloud MNIST 2D [18, 34] consists of 60,000 training samples and 10,000 testing samples. Each sample is a 2-dimensional point cloud derived from an image in the original MNIST dataset. The sets have various cardinalities in the range of $|X_i| \in [34, 351]$.

ModelNet40 [68] contains 3-dimensional point clouds converted from 12,311 CAD models in 40 common object categories. We used the official split with 9,843 samples for training and 2,468 samples for testing. We sample N_i points uniformly and randomly from the mesh faces of each object, where $N_i = \lfloor n_i \rfloor$, $n_i \sim \mathcal{N}(512, 64)$. To avoid any orientation bias, we randomly rotate the point clouds by $\rho \in [0, 45]$ degrees around the x-axis. Finally, we normalize each sample to fit within the unit cube to avoid any scale bias and to enforce the methods to rely on shapes.

The Oxford Buildings Dataset [49] has 5,062 images containing eleven different Oxford landmarks. Each land-

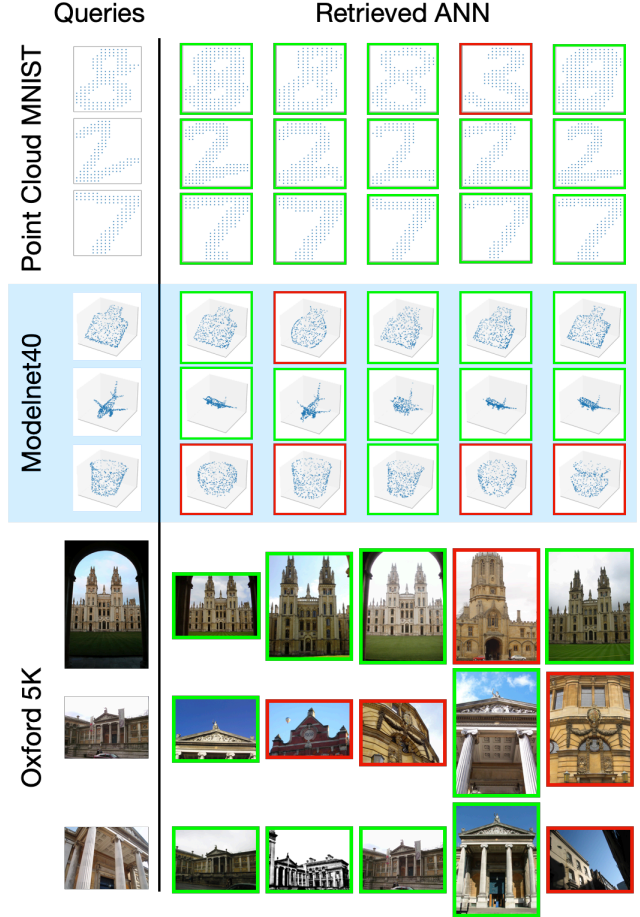


Figure 2. Example sample retrievals.

mark has five corresponding queries leading to 55 queries over which an object retrieval system can be evaluated. In our experiments, we use a pretrained VGG16 [57] on ImageNet1k [54] as a feature extractor, and use the features in the last convolutional layer as a set representation for an input image. We resize the images without cropping, which leads to varied input size images, and therefore gives set representations with varied cardinalities. We further perform a dimensionality reduction on these features to obtain sets of features in an $(d = 8)$ -dimensional space.

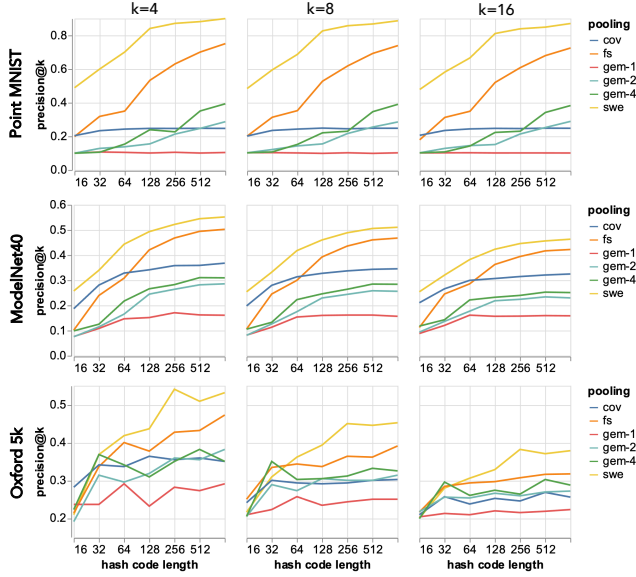


Figure 3. Sensitivity to hash code length.

5.3. Results

For all datasets, we first calculate the set-2-vector embeddings for all baselines and SWE. Then, we apply Locality-Sensitive Hashing (LSH) to the embedded sets and report **Precision@k** and accuracy (based on majority voting) for all the approaches on the test sets. We use the FAISS library [24], developed by Facebook AI Research, for LSH. For all methods, we use a hash code length of 1024, and we report our results for $k = 4, 8$, and 16. For SLOSH, we consider three different settings for the number of slices, namely, $L < d$, $L = d$, and $L > d$. We repeat the experiments five times per method and report the mean Precision@k and accuracy in Table 1. In all experiments, the best hyper-parameters are selected for each approach based on cross validation. **We see that SLOSH provides a consistent lead on all datasets, especially, when $L > d$.** Figure 2 provide set retrieval examples on the three data sets. Next, we provide a sensitivity analysis of our approach.

Sensitivity to code length. For all datasets, we study the sensitivity of the different embeddings to the hashing code-length. We vary the code length from 16 to 1024, and report the average of Precision@k over five runs. Figure 3 shows the outcome of this study. It can be seen that methods that encode higher statistical orders of the underlying distribution gain performance as a function of the hash code length. In addition, we see a strong performance gap between SWE and other set-2-vector approaches, which points at the more descriptive nature of our proposed method.

Sensitivity to the number of slices. Next, we study the sensitivity of SLOSH to the choice of number of slices, L , for the three datasets. We measure the average Precision@k over five runs for various number of slices and for different

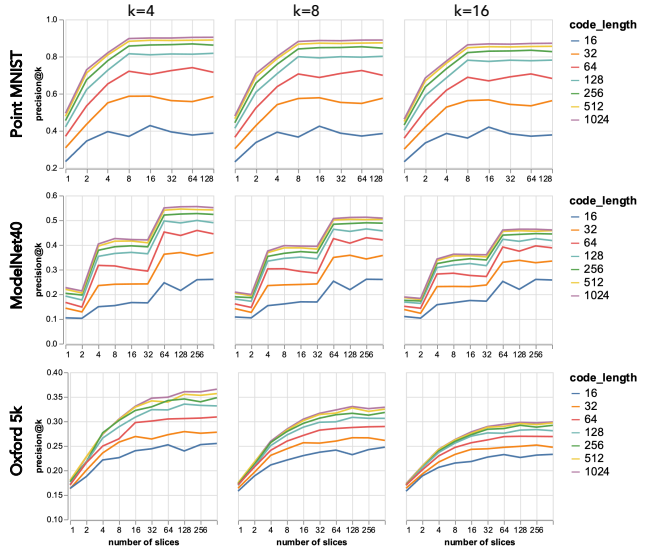


Figure 4. Sensitivity to number of slices.

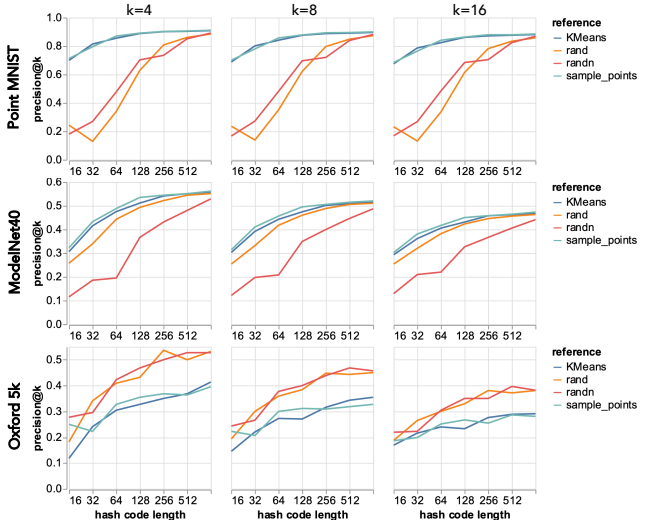


Figure 5. Sensitivity to reference functions.

code-lengths and report our results in Figure 4. As expected we see a performance increase with respect to increasing L .

Sensitivity to the reference. Finally, we perform a study on the sensitivity to the choice of the reference set, X_0 . We measure the performance of SLOSH on the three datasets and for various code-lengths, when the reference set is: 1) calculated using K-Means on the elements of all sets, 2) a random set from the dataset, 3) sampled from a uniform distribution, and 4) sampled from the normal distribution. We see that SLOSH’s performance depends on the reference choice. However, we point out that our formulation implies that as $L \rightarrow \infty$ the embedding becomes independent of the choice of the reference, and the observed sensitivity could be due to using finite L .

6. Conclusion

We described a novel data-independent approach for Approximate Nearest Neighbor (ANN) search on set-structured data, with applications in set retrieval. We treat set elements as samples from an underlying distribution, and embed sets into a vector space in which the Euclidean distance approximates the Sliced-Wasserstein (SW) distance between the input distributions. We show that for a set X with cardinality $|X| = N$, our framework requires $\mathcal{O}(LN(d + \log(N)))$ (sequential processing) or $\mathcal{O}(N(d + \log(N)))$ (parallel processing) calculations to obtain the embedding. We then use Locality Sensitive Hashing (LSH) for fast retrieval of nearest sets in our proposed embedding. We demonstrate a significant boost over other data-independent approaches including Generalized Mean (GeM) on three different set retrieval tasks, namely, Point Cloud MNIST, ModelNet40, and the Oxford Buildings datasets. Finally, our proposed method is readily extendable to data-dependent settings by allowing optimization on the slices and the reference set.

7. Acknowledgements

This research was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR00112190132.

References

- [1] Dinesh Acharya, Zhiwu Huang, Danda Pani Paudel, and Luc Van Gool. Covariance pooling for facial expression recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 367–374, 2018. [6](#)
- [2] David Alvarez Melis and Nicolo Fusi. Geometric dataset distances via optimal transport. *Advances in Neural Information Processing Systems*, 33, 2020. [3](#)
- [3] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*, pages 459–468. IEEE, 2006. [1](#), [2](#)
- [4] Alexandr Andoni, Piotr Indyk, Huy L Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1018–1028. SIAM, 2014. [2](#)
- [5] Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. Approximate nearest neighbor search in high dimensions. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 3287–3318. World Scientific, 2018. [1](#)
- [6] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 793–801, 2015. [1](#), [2](#)
- [7] Relja Arandjelovic and Andrew Zisserman. All about vlad. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1578–1585, 2013. [2](#)
- [8] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017. [3](#)
- [9] Yogesh Balaji, Rama Chellappa, and Soheil Feizi. Normalized wasserstein for mixture distributions with applications in adversarial learning and domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6500–6508, 2019. [3](#)
- [10] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. [1](#)
- [11] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. [2](#)
- [12] Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and Radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015. [5](#)
- [13] Nicolas Bonnotte. *Unidimensional and evolution methods for optimal transportation*. PhD thesis, Université Paris 11, France, 2013. [2](#), [3](#)
- [14] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002. [4](#)
- [15] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865, 2016. [3](#)
- [16] Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 447–463, 2018. [3](#)
- [17] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004. [1](#), [2](#), [3](#), [4](#)
- [18] Cristian Garcia. Point cloud mnist 2d, 2020. [6](#), [7](#)
- [19] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19:513–520, 2006. [2](#)
- [20] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017. [3](#)
- [21] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998. [1](#), [2](#), [3](#), [4](#)

- [22] Tony Jebara, Risi Kondor, and Andrew Howard. Probability product kernels. *The Journal of Machine Learning Research*, 5:819–844, 2004. 2
- [23] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3304–3311. IEEE, 2010. 2
- [24] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017. 8
- [25] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *European conference on computer vision*, pages 685–701. Springer, 2016. 2
- [26] Haim Kaplan and Jay Tenenbaum. Locality sensitive hashing for set-queries, motivated by group recommendations. In *17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020. 3
- [27] Soheil Kolouri, Navid Naderializadeh, Gustavo K. Rohde, and Heiko Hoffmann. Wasserstein embedding for graph learning. In *International Conference on Learning Representations*, 2021. 2, 3
- [28] Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo Rohde. Generalized sliced wasserstein distances. In *Advances in Neural Information Processing Systems*, pages 261–272, 2019. 3
- [29] Soheil Kolouri, Se Rim Park, and Gustavo K. Rohde. The Radon cumulative distribution transform and its application to image classification. *Image Processing, IEEE Transactions on*, 25(2):920–934, 2016. 4
- [30] Soheil Kolouri, Se Rim Park, Matthew Thorpe, Dejan Slepcev, and Gustavo K Rohde. Optimal mass transport: Signal processing and machine-learning applications. *IEEE Signal Processing Magazine*, 34(4):43–59, 2017. 2, 3, 4
- [31] Soheil Kolouri, Phillip E. Pope, Charles E. Martin, and Gustavo K. Rohde. Sliced Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2019. 3, 5
- [32] Soheil Kolouri, Yang Zou, and Gustavo K Rohde. Sliced-Wasserstein kernels for probability distributions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4876–4884, 2016. 2, 5
- [33] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *2009 IEEE 12th international conference on computer vision*, pages 2130–2137. IEEE, 2009. 2
- [34] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 6, 7
- [35] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10285–10295, 2019. 3
- [36] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019. 1, 2
- [37] Alexander Levine and Soheil Feizi. Wasserstein smoothing: Certified robustness against wasserstein adversarial attacks. In *International Conference on Artificial Intelligence and Statistics*, pages 3938–3947. PMLR, 2020. 3
- [38] Antoine Liutkus, Umut Simsekli, Szymon Majewski, Alain Durmus, and Fabian-Robert Stöter. Sliced-wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In *International Conference on Machine Learning*, pages 4104–4113. PMLR, 2019. 3, 5
- [39] Antoine Liutkus, Umut Şimşekli, Szymon Majewski, Alain Durmus, and Fabian-Robert Stöter. Sliced-Wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In *International Conference on Machine Learning*, 2019. 5
- [40] Grégoire Mialon, Dexiong Chen, Alexandre d’Aspremont, and Julien Mairal. A trainable optimal transport embedding for feature aggregation and its relationship to attention. In *International Conference on Learning Representations*, 2021. 3
- [41] Caroline Moosmüller and Alexander Cloninger. Linear optimal transport embedding: Provable fast wasserstein distance computation and classification for nonlinear problems. *arXiv preprint arXiv:2008.09165*, 2020. 4
- [42] Krikamol Muandet, Kenji Fukumizu, Francesco Dinuzzo, and Bernhard Schölkopf. Learning from distributions via support measure machines. In *Proceedings of the 25th International Conference on Neural Information Processing Systems-Volume 1*, pages 10–18, 2012. 2
- [43] Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In *International Conference on Learning Representations*, 2019. 2
- [44] Kimia Nadjahi, Alain Durmus, Lénaïc Chizat, Soheil Kolouri, Shahin Shahrampour, and Umut Şimşekli. Statistical and topological properties of sliced probability divergences. In *Advances in Neural Information Processing Systems*, 2020. 5
- [45] Parth Nagarkar and K Selçuk Candan. Pslsh: An index structure for efficient execution of set queries in high-dimensional spaces. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 477–486, 2018. 3
- [46] Se Rim Park, Soheil Kolouri, Shinjini Kundu, and Gustavo K Rohde. The cumulative distribution transform and linear pattern classification. *Applied and Computational Harmonic Analysis*, 45(3):616–641, 2018. 4, 11
- [47] François-Pierre Paty and Marco Cuturi. Subspace robust wasserstein distances. In *International Conference on Machine Learning*, 2019. 3
- [48] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *arXiv preprint arXiv:1803.00567*, 2018. 2, 3

- [49] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007. 6, 7
- [50] Barnabás Póczos and Jeff Schneider. Nonparametric estimation of conditional information and divergences. In *Artificial Intelligence and Statistics*, pages 914–923. PMLR, 2012. 2
- [51] Barnabás Póczos, Liang Xiong, and Jeff Schneider. Nonparametric divergence estimation with applications to machine learning on distributions. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 599–608, 2011. 2
- [52] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *Scale Space and Variational Methods in Computer Vision*, pages 435–446. Springer, 2012. 3, 5
- [53] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1655–1668, 2018. 2, 6
- [54] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 7
- [55] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. Nearest-neighbor methods in learning and vision. *IEEE Trans. Neural Networks*, 19(2):377, 2008. 1
- [56] Mohammad Shifat-E-Rabbi, Xu Wang Yin, Abu Hasnat Mohammad Rubaiyat, Shiyong Li, Soheil Kolouri, Akram Aldroubi, Jonathan M Nichols, and Gustavo K Rohde. Radon cumulative distribution transform subspace modeling for image classification. *Journal of Mathematical Imaging and Vision*, 63(9):1185–1203, 2021. 4
- [57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 7
- [58] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifying some distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018. 3
- [59] Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. Wasserstein weisfeiler-lehman graph kernels. *Advances in Neural Information Processing Systems*, 32:6439–6449, 2019. 3
- [60] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018. 3
- [61] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008. 2, 3
- [62] Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A Osborne. On the limitations of representing functions on sets. In *International Conference on Machine Learning*, pages 6487–6494. PMLR, 2019. 1
- [63] Ingo Wald and Vlastimil Havran. On building fast kd-trees for ray tracing, and on doing that in $O(n \log n)$. In *2006 IEEE Symposium on Interactive Ray Tracing*, pages 61–69. IEEE, 2006. 1
- [64] Qilong Wang, Jiangtao Xie, Wangmeng Zuo, Lei Zhang, and Peihua Li. Deep cnns meet global covariance pooling: Better representation and generalization. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 2, 6
- [65] Wei Wang, Dejan Slepčev, Saurav Basu, John A Ozolek, and Gustavo K Rohde. A linear optimal transportation framework for quantifying and visualizing variations in sets of images. *International journal of computer vision*, 101(2):254–269, 2013. 4
- [66] Eric Wong, Frank Schmidt, and Zico Kolter. Wasserstein adversarial examples via projected sinkhorn iterations. In *International Conference on Machine Learning*, pages 6808–6817. PMLR, 2019. 3
- [67] Kaiwen Wu, Allen Wang, and Yaoliang Yu. Stronger and faster wasserstein adversarial attacks. In *International Conference on Machine Learning*, pages 10377–10387. PMLR, 2020. 3
- [68] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 6, 7
- [69] Liang Xiong and Jeff Schneider. Learning from point sets with observational bias. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, pages 898–906, 2014. 2
- [70] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J Smola. Deep sets. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3394–3404, 2017. 1, 2
- [71] Yan Zhang, Jonathon Hare, and Adam Prügel-Bennett. Fspool: Learning set representations with featurewise sort pooling. *arXiv preprint arXiv:1906.02795*, 2019. 6, 7
- [72] Yan Zhang, Jonathon Hare, and Adam Prügel-Bennett. Fspool: Learning set representations with featurewise sort pooling. In *International Conference on Learning Representations*, 2020. 2

8. Supplementary Materials

8.1. Proofs

Here we include the proof for the C1 and C2 conditions covered in Section 4. Recall that μ_i represents a probability measure, μ_i^θ represents $g_{\theta\#}\mu_i$, where $g_\theta : \mathbb{R}^d \rightarrow \mathbb{R}$ with some regularity constraints, and that we define the cumulative distribution transform (CDT) [46] of μ_i^θ as

$$\phi_\theta(\mu_i) := (T_i^\theta - id),$$

where T_i^θ is the Monge map/coupling, and id denotes the identity function. For a fixed θ , here we prove that $\phi^\theta(\mu_i)$ satisfies the following conditions:

Method	Complexity
Gem-p	$\mathcal{O}(Np^2d)$
Cov.	$\mathcal{O}(Nd^2)$
FSPool	$\mathcal{O}(Nd \log N)$
SLOSH	$\mathcal{O}(LN(d + \log N))$

Table 2. Computational complexities.

C1. The weighted ℓ_2 -norm of the embedded slice, $\phi_\theta(\mu_i)$, satisfies:

$$\begin{aligned} \|\phi_\theta(\mu_i)\|_{\mu_0^\theta, 2} &= \left(\int_{\mathbb{R}} \|\phi_\theta(\mu_i(t))\|_2^2 d\mu_0^\theta(t) \right)^{\frac{1}{2}} \\ &= \mathcal{W}_2(\mu_i^\theta, \mu_0^\theta), \end{aligned}$$

Proof. We start by writing the squared distance:

$$\begin{aligned} \|\phi_\theta(\mu_i)\|_{\mu_0^\theta, 2}^2 &= \int_{\mathbb{R}} \|\phi_\theta(\mu_i(t))\|_2^2 d\mu_0^\theta(t) \\ &= \int_{\mathbb{R}} \|T_i^\theta(t) - t\|_2^2 d\mu_0^\theta(t) \\ &= \int_{\mathbb{R}} \|(F_{\mu_i^\theta}^{-1} \circ F_{\mu_0^\theta})(t) - t\|_2^2 d\mu_0^\theta(t) \\ &= \int_0^1 \|F_{\mu_i^\theta}^{-1}(\tau) - F_{\mu_0^\theta}^{-1}(\tau)\|_2^2 d\tau \\ &= \mathcal{W}_2^2(\mu_i^\theta, \mu_0^\theta) \end{aligned}$$

where we used the definition of the one-dimensional Monge map, $T_i^\theta = F_{\mu_i^\theta}^{-1} \circ F_{\mu_0^\theta}$, and the change of variable $\tau = F_{\mu_0^\theta}(t)$. The corollary, $\|\phi_\theta(\mu_0)\|_{\mu_0^\theta, 2} = 0$, is trivial as $\mathcal{W}_2^2(\mu_0^\theta, \mu_0^\theta) = 0$. \square

C2. The weighted ℓ_2 distance satisfies:

$$\|\phi_\theta(\mu_i) - \phi_\theta(\mu_j)\|_{\mu_0^\theta, 2} = \mathcal{W}_2(\mu_i^\theta, \mu_j^\theta). \quad (18)$$

Proof. Similar to the previous proof:

$$\begin{aligned} \|\phi_\theta(\mu_i) - \phi_\theta(\mu_j)\|_{\mu_0^\theta, 2}^2 &= \int_{\mathbb{R}} \|\phi_\theta(\mu_i(t)) - \phi_\theta(\mu_j(t))\|_2^2 d\mu_0^\theta(t) \\ &= \int_{\mathbb{R}} \|T_i^\theta(t) - T_j^\theta(t)\|_2^2 d\mu_0^\theta(t) \\ &= \int_{\mathbb{R}} \|(F_{\mu_i^\theta}^{-1} \circ F_{\mu_0^\theta})(t) - (F_{\mu_j^\theta}^{-1} \circ F_{\mu_0^\theta})(t)\|_2^2 d\mu_0^\theta(t) \\ &= \int_0^1 \|F_{\mu_i^\theta}^{-1}(\tau) - F_{\mu_j^\theta}^{-1}(\tau)\|_2^2 d\tau = \mathcal{W}_2^2(\mu_i^\theta, \mu_j^\theta) \end{aligned}$$

where again we used the definition of the one-dimensional Monge map, $T_i^\theta = F_{\mu_i^\theta}^{-1} \circ F_{\mu_0^\theta}$, and the change of variable $\tau = F_{\mu_0^\theta}(t)$. \square

As a corollary of C1 and C2 we have:

$$\begin{aligned} \mathcal{GSW}_2^2(\mu_i, \mu_j) &= \int_{\Omega_\theta} \mathcal{W}_2^2(\mu_i^\theta, \mu_j^\theta) d\sigma(\theta) = \\ &= \int_{\Omega_\theta} \|\phi_\theta(\mu_i) - \phi_\theta(\mu_j)\|_{\mu_0^\theta, 2}^2 d\sigma(\theta) \end{aligned}$$

8.2. Computational complexities

For the sake of completeness, here we include the computational complexities of the baseline methods used in our paper. In Table 2, we provide the computational complexity of embedding a set $X = \{x_n \in \mathbb{R}^d\}_{n=1}^N$ into a vector space.