# SLoSH: Set Locality Sensitive Hashing via Sliced-Wasserstein Embeddings

**Yuzhe Lu**[*1], **Xinran Liu**[*2], **Andrea Soltoggio**[3], **Soheil Kolouri**[2]
[1]Carnegie Mellon University [2]Vanderbilt University [3]Loughborough University
yuzhelu@cs.cmu.edu, xinran.liu@vanderbilt.edu
a.soltoggio@lboro.ac.uk, 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 a non-parametric, data-independent, and efficient learning algorithm from set-structured data using optimal transport and approximate nearest neighbor (ANN) solutions, particularly locality-sensitive hashing. We consider the problem of set retrieval from an input set query. This retrieval problem requires 1) an efficient mechanism to calculate the distances/dissimilarities between sets and 2) an appropriate data structure for a fast nearest-neighbor search. To that end, we propose to use Sliced-Wasserstein embedding as a computationally efficient "set-2-vector" operator 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), Covariance Pooling, and Wasserstein embedding and show consistent improvement in retrieval results, both in terms of accuracy and computational efficiency.*

## 1. Introduction

The nearest neighbor search problem is at the heart of many non-parametric 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) [16, 20], 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, most 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 [12, 29, 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, 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 the Hungarian algorithm, Wasserstein distances,

[*]Equal contribution

Chamfer loss, etc. These approaches are, at best, quadratic and, at worst, cubic in the set cardinalities.

The second family of approaches relies 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) [11,18,21,32,41,50,51,69]. 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 is the case for global average/max/sum/covariance pooling, variations of Janossy pooling [42], and variations of Wasserstein embedding [26], 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 transport-based 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 $\mathcal{X}$. We propose the composition of the Sliced Wasserstein Embedding (SWE) [28,43] and Locality Sensitive Hashing (LSH) [16, 20], denoted as Set Locality Sensitive Hashing (SLoSH), as a fast set retrieval mechanism with theoretical guarantees. We treat sets as empirical distributions and use SWE to embed sets in a vector space in which the Euclidean 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. To summarize, we devise an approximate nearest neighbor retrieval algorithm for set-structure data based on the sliced-Wasserstein distance, denoted as SLoSH, provide theoretical bounds for the proposed method, and demonstrate efficient computational complexity and superior retrieval performance.

## 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).

Non-parametric learning from set-structured data remains a relatively understudied area, which is often limited to the common set-2-vector operators used as global pooling mechanisms in modern deep learning architectures. In particular, global average/max/sum and covariance pooling [64] could be considered as the simplest such processes. Generalized Mean (GeM) [53] is another set-2-vector mechanism commonly used in image retrieval applications, which captures higher statistical moments of the underlying distributions. Among other notable approaches are VLAD [7, 22], CroW [24], FSPool [72], Wasserstein Embedding [26], and SWE [43].

**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 [20] and extended in [16], 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].

We are interested in nearest neighbor retrieval for sets, and propose to extend LSH to enable its application to set retrieval. While there has been a few recent works [25, 45] on the topic of LSH for set queries, our proposed approach significantly differs from these works. In contrast to [25, 45] and following the work of [43] 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. Notably, while other ANN methods could also be used in our experiments, LSH allows us to quantify the probability of having the same hash codes for two sets as the SW distance between their distribution.

**Wasserstein distances:** Rooted in the optimal transportation problem [29, 48, 61], Wasserstein distances 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 [52] and subspace robust Wasserstein distances [47], have been extensively studied in the context of deep generative modeling [8, 19, 30, 30, 38, 60], domain adaptation [9, 14, 15, 35], transfer learning [2, 37], adversarial attacks [66, 67] , and adversarial robustness [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, [59] proposed to compare node embeddings of two graphs (treated as sets) via the Wasserstein distance. Later, [39] and [26] propose Wasserstein embedding frameworks for extracting fixed-dimensional representations from set-structured data, and [43] extend this framework to Sliced Wasserstein Embedding (SWE) in a parametric learning setting as a pooling layer in deep neural networks for sets. Here, we further extend this direction and propose SWE as a computationally efficient approach that allows us to perform data-independent non-parametric learning from sets.

## 3. Problem Formulation and Method

### 3.1. Sliced-Wasserstein Embedding

The idea of Sliced-Wasserstein Embeddings (SWE) is rooted in Linear Optimal Transport [29, 40, 65] and was first introduced in the context of pattern recognition from 2D probability density functions (e.g., images) [28] and more recently in [56]. Naderializadeh et al. [43] extend this framework to d-dimensional distributions. Consider a set of probability measures $\{\mu_i\}_{i=1}^{I}$, where we use $\mu_i$ to represent the i'th set $X_i = \{x_n^i \in \mathbb{R}^d\}_{n=0}^{N_i-1}$, i.e., $\mu_i(x) = \frac{1}{N_i}\sum_{n=0}^{N_i-1}\delta(x - x_n^i)$ where $\delta$ is the Dirac function. At a high level, SWE can be thought as a permutation invariant set-2-vector operator, $\phi$, such that:

$$\|\phi(\mu_i) - \phi(\mu_j)\|_2 = \mathcal{SW}_2(\mu_i, \mu_j). \tag{1}$$

Given a defining function [27], $g_\theta : \mathbb{R}^d \to \mathbb{R}$, a slice of $\mu_i$ with respect to $g_\theta$ can be written as $\mu_i^\theta := g_{\theta\#}\mu_i$. In this paper, and without the loss of generality, we use $g_\theta(x) = x \cdot \theta$. Moreover, let $\mu_0$ denote a reference measure, with $\mu_0^\theta$ being its corresponding slice. The optimal transport map, i.e., Monge map, between $\mu_i^\theta$ and $\mu_0^\theta$ is written as,

$$T_i^\theta = F_{\mu_i^\theta}^{-1} \circ F_{\mu_0^\theta}, \tag{2}$$

where $F_{\mu_i^\theta}^{-1}$ is the quantile function of $\mu_i^\theta$, the inverse of the cumulative distribution function (CDF). Let $id$ denote the identity function, the cumulative distribution transform (CDT) [46] of $\mu_i^\theta$ is defined as,

$$\phi_\theta(\mu_i) := (T_i^\theta - id), \tag{3}$$

For a fixed $\theta$, $\phi^\theta(\mu_i)$ satisfies the following conditions (see supplementary material for the proof):

**C1.** The weighted $\ell_2$-norm of the embedded slice, $\phi_\theta(\mu_i)$, satisfies:

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

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

**C2.** The distance between two embedded slices satisfies:

$$\|\phi_\theta(\mu_i) - \phi_\theta(\mu_j)\|_{\mu_0^\theta,2} = \mathcal{W}_2(\mu_i^\theta, \mu_j^\theta). \tag{4}$$

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

$$\mathcal{SW}_2(\mu_i, \mu_j) = \left(\int_{\mathbb{S}^{d-1}} \|\phi_\theta(\mu_i) - \phi_\theta(\mu_j)\|_{\mu_0^\theta,2}^2 d\theta\right)^{\frac{1}{2}} \tag{5}$$

For probability measure $\mu_i$, we then define the mapping to the embedding space via, $\phi(\mu_i) := \{\phi_\theta(\mu_i) \mid \theta \in \mathbb{S}^{d-1}\}$. Next, we describe the implementation considerations for SWE in more detail.

### 3.2. Monte Carlo Integral Approximation

The SW distance in Eq. (5) relies on integration on $\mathbb{S}^{d-1}$, which cannot be directly calculated. Following the common practice in the literature [30, 52], we approximate the integration on $\theta$ via a Monte-Carlo (MC) sampling of $\mathbb{S}^{d-1}$. Let $\Theta_L = \{\theta_l\}_{l=1}^{L}$ denote a set of $L$ unit vectors sampled independently and uniformly from $\mathbb{S}^{d-1}$. 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 can then be written as:

$$\widehat{\mathcal{SW}}_{2,L}^2(\mu_i, \mu_j) = \frac{1}{LM}\sum_{l=1}^{L}\|\phi_{\theta_l}(\mu_i) - \phi_{\theta_l}(\mu_j)\|_2^2. \tag{6}$$

Finally, our SWE embedding is calculated via:

$$\phi(\mu_i) = [\frac{\phi_{\theta_1}(\mu_i)}{\sqrt{LM}}; ...; \frac{\phi_{\theta_L}(\mu_i)}{\sqrt{LM}}] \in \mathbb{R}^{LM\times 1}, \tag{7}$$

which satisfies:

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

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

$$\mathbb{E}[|\widehat{\mathcal{SW}}_{2,L}^2(\mu_i, \mu_j) - \mathcal{SW}_2^2(\mu, \nu)|] \leq \sqrt{\frac{\text{var}(\mathcal{W}_2^2(\mu_i^\theta, \mu_j^\theta))}{L}} \tag{8}$$

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.

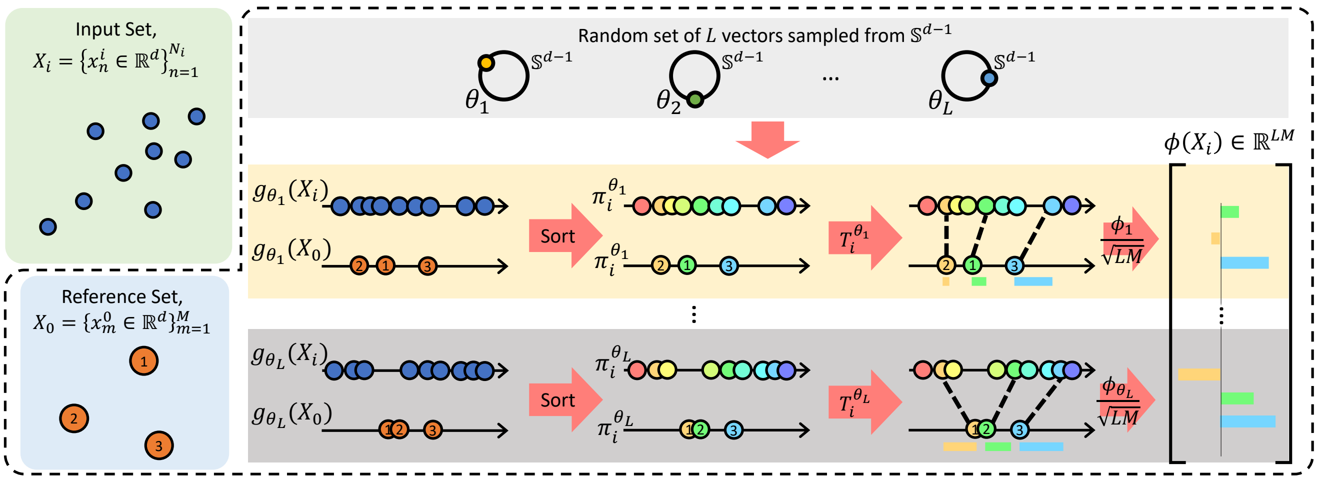
Input Set,
$X_i = \{x_n^i \in \mathbb{R}^d\}_{n=1}^{N_i}$
Reference Set,
$X_0 = \{x_m^0 \in \mathbb{R}^d\}_{m=1}^{M}$
Random set of $L$ vectors sampled from $\mathbb{S}^{d-1}$
$\theta_1$
$\theta_2$
$\theta_L$
$\phi(X_i) \in \mathbb{R}^{LM}$
$g_{\theta_1}(X_i)$
$g_{\theta_1}(X_0)$
Sort
$\pi_i^{\theta_1}$
$T_i^{\theta_1}$
$\frac{\phi_1}{\sqrt{LM}}$
$g_{\theta_L}(X_i)$
$g_{\theta_L}(X_0)$
Sort
$\pi_i^{\theta_L}$
$T_i^{\theta_L}$
$\frac{\phi_{\theta_L}}{\sqrt{LM}}$

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. (9). Finally, the embedding is obtained by weighting and concatenating $\phi_{\theta_l}$s. The Euclidean distance between two embedded sets is equal to their corresponding $\widehat{\mathcal{SW}}_{2,L}$ distance/dissimilarity measure, i.e., $\|\phi(X_i) - \phi(X_j)\|_2 = \widehat{\mathcal{SW}}_{2,L}(X_i, X_j) \approx \mathcal{SW}_2(X_i, X_j)$.

## 3.3. 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 $g_\theta$ 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 $\pi_i$ and $\pi_0$ denote the permutation indices (obtained from argsort). Also, let $\pi_0^{-1}$ denote the ordering that permutes the sorted set back to the original ordering. Then we numerically calculate the Monge coupling $T_i^\theta$ via:

$$T_i^\theta[m] = F_{\mu_i^\theta}^{-1}\left(\frac{\pi_0^{-1}[m]+1}{M}\right) \tag{9}$$

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:

$$\phi_\theta(X_i)[m] = (T_i^\theta[m] - g_\theta(x_m^0)) \tag{10}$$

and repeat this process for $\theta \in \Theta$, while we emphasize that this process can be parallelized. The final embedding is achieved via weighting and concatenating $\phi_{\theta_l}$s as in Eq. (7), 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 the process. Lastly, the SWE's computational complexity for a set with cardinality $|X| = N$ is $\mathcal{O}(LN(d + logN))$, 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 $\mathcal{O}(LNlog(N))$ is the sorting and interpolation cost.

**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 $\Theta_L = \{\theta_l \sim \mathcal{U}_{\mathbb{S}^{d-1}}\}_{l=1}^{L}$
  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. (9))
  end for
  Calculate the embedding $\phi(X_i) \in \mathbb{R}^{M\times L}$ (Eq. (7))
  return $\phi(X_i)$
end procedure
```

## 3.4. SLoSH

Our proposed Set Locality Sensitive Hashing (SLoSH) leverages the 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 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 $\widehat{\mathcal{SW}}_{2,L}(X_i, X_j) \leq R$, then $Pr[h(\phi(X_i)) = h(\phi(X_j))] \geq P_1$, and
- If $\widehat{\mathcal{SW}}_{2,L}(X_i, X_j) > cR$, then $Pr[h(\phi(X_i)) = h(\phi(X_j))] \leq P_2$

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

## 4. Experiments

We evaluated SLoSH against other set-2-vector approaches using Generalized Mean (GeM) pooling [53], Global Covariance (Cov) pooling [64], Featurewise Sort Pooling (FSPool) [71], and Wasserstein Embedding [26] 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. Interstingly, FSPool can be thought as a special case of our SWE embedding where $L = d$ and $\Theta_L$ is chosen as the identity matrix. To evaluate these methods, we tested all approaches on point cloud MNIST dataset (2D) [17, 34], ModelNet40 dataset (3D) [68], and the Oxford Buildings dataset (8D) [49]. We conducted experiments on CPU (10 cores, 16 GB memory) and macOS 12.4 operating system.

### 4.1. 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, $k \in \{1, 2, ...d\}$. 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:

$$[f^{(p)}]_k = \left( \frac{1}{N_i} \sum_{n=1}^{N_i} ([x_n^i]_k)^p \right)^{\frac{1}{p}} \tag{11}$$

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

$$C = \frac{1}{N_i - 1} \sum_{n=1}^{N_i} (x_n^i - \overline{\mu}_i)(x_n^i - \overline{\mu}_i)^T, \tag{12}$$

where $\overline{\mu}_i = \frac{1}{N_i} \sum_{n=1}^{N_i} x_n^i$. 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), $C_\lambda = C + \lambda trace(C)\mathbf{I}$ where $\lambda$ is a regularization hyper-parameter and $\mathbf{I}$ is the identity matrix. Covariance pooling then uses $\phi_{Cov}(X_i) = \text{flatten}(C_\lambda)$.

**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, $[X_i]_k$:

$$f = [Sorted([X_i]_1), ..., Sorted([X_i]_d)] \in \mathbb{R}^{N_i \times d} \tag{13}$$

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 altogether 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.

**Wasserstein Embedding (WE)** was described in [65] and the follow-up works [13,26,31] as an isometric Hilbertian embedding of probability measures such that the Euclidean distance between the embedded vectors approximates the 2-Wasserstein distance $\mathcal{W}_2$. Assume the input set $X_i = \{x_n^i \in \mathbb{R}^d\}_{n=0}^{N_i-1}$ are i.i.d. samples from the probability distribution $p_i$, and let us define $p_0$ as the reference distribution with samples $X_0 = \{x_m^0 \in \mathbb{R}^d\}_{m=0}^{M}$. Wasserstein embedding for $X_i$ can be computed by

$$WE(X_i) = (F_i - X_0)/\sqrt{M} \in \mathbb{R}^{N \times d} \tag{14}$$

Table 1. Results of baselines and our approach on three set retrieval tasks (full table with std in supplemental material). For each method, we show their time complexity to embed sets and their performance measured by Precision@k and accuracy. Compared to GeM-N, Cov and FSPool, SLoSH significantly improves performance. Compared to WE, SLoSH achieves comparable performance on 2/3 tasks while being significantly faster, which is critical for ANN solutions. This accuracy-efficiency trade-off is better illustrated in Figure 2.

| | | Point MNIST 2D | | | ModelNet 40 | | | Oxford 5k | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Precision@k/Accuracy | | | Precision@k/Accuracy | | | Precision@k/Accuracy | | |
| Methods | Complexities | k=4 | k=8 | k=16 | k=4 | k=8 | k=16 | k=4 | k=8 | k=16 |
| GeM-1 (GAP) | $\mathcal{O}(Nd)$ | 0.10/0.11 | 0.10/0.10 | 0.10/0.10 | 0.14/0.17 | 0.14/0.19 | 0.14/0.21 | 0.29/0.35 | 0.25/0.31 | 0.22/0.29 |
| GeM-2 | $\mathcal{O}(Nd)$ | 0.29/0.32 | 0.29/0.35 | 0.29/0.37 | 0.30/0.34 | 0.28/0.36 | 0.25/0.37 | 0.38/0.53 | 0.31/0.40 | 0.27/0.38 |
| GeM-4 | $\mathcal{O}(Nd)$ | 0.39/0.45 | 0.39/0.47 | 0.38/0.49 | 0.34/0.39 | 0.31/0.39 | 0.28/0.39 | 0.09/0.09 | 0.09/0.09 | 0.09/0.09 |
| Cov | $\mathcal{O}(Nd^2)$ | 0.25/0.26 | 0.25/0.28 | 0.25/0.28 | 0.45/0.51 | 0.43/0.52 | 0.41/0.52 | 0.35/0.55 | 0.30/0.37 | 0.26/0.33 |
| FSPool | $\mathcal{O}(Nd\log N)$ | 0.75/0.80 | 0.74/0.81 | 0.72/0.81 | 0.51/0.58 | 0.48/0.58 | 0.44/0.58 | 0.43/0.50 | 0.36/0.53 | 0.36/0.44 |
| WE | $\mathcal{O}(N^3\log N)$ | **0.89/0.92** | **0.88/0.92** | **0.86/0.92** | **0.71/0.76** | **0.68/0.76** | **0.64/0.75** | **0.54/0.70** | **0.47/0.68** | **0.39/0.61** |
| SLoSH ($L = d$) | $\mathcal{O}(Nd\log N))$ | 0.78/0.82 | 0.76/0.83 | 0.74/0.82 | 0.40/0.46 | 0.38/0.47 | 0.35/0.46 | 0.43/0.54 | 0.36/0.53 | 0.30/0.45 |
| SLoSH ($L > d$) | $\mathcal{O}(NL\log N)$ | **0.89/0.92** | **0.88/0.92** | **0.86/0.92** | **0.63/0.68** | **0.59/0.67** | **0.55/0.65** | **0.52/0.69** | **0.45/0.67** | **0.37/0.61** |

where $F_i$ is the Monge map between $p_i$ and $p_0$, approximated via the barycentric projection from the optimal transport plan. Note that the optimal transport plan is solved using linear programming with complexity $\mathcal{O}(N^3 \log(N))$.

## 4.2. Implementation Details

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 (from majority voting) for all approaches on test sets. We use the LSH implementation from FAISS library [23]. For all methods, we use a hash code length of 1024, and we report our results for different number of nearest neighbors $k = 4, 8,$ and 16. For SLoSH, we consider two different settings for the number of slices, namely, $L = d$, 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 hyperparameters are selected based on the sensitivity analysis on the validation set in Section 4.5. To improve reproducibility, we submitted code in our supplemental material.

## 4.3. Datasets

Next, we cover details of the three datasets utilized in our experiments to demonstrate the superiority of SLoSH.

**Point Cloud MNIST 2D** [17] 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 [34]. 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 rotate the point clouds by 45 degrees

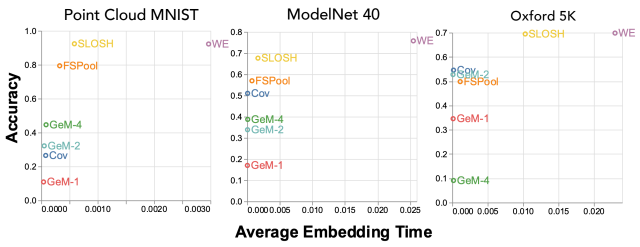

Figure 2. We present a wall clock analysis to demonstrate the trade-off between the accuracy and computational efficiency of different methods. Compared to WE, SLoSH offers comparable performance but is faster. Compared to other baselines, SLoSH improves accuracy by a large margin while adding little computational time, except when the data dimension is high (Oxford 5K).

around the x-axis. Finally, we normalize each sample to fit within the unit cube to avoid any scale bias.

**The Oxford Buildings Dataset** [49] has 5,062 images containing eleven different Oxford landmarks. Each landmark has five corresponding queries leading to 55 queries over which an object retrieval system can be evaluated. In our experiments, we used 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 dimension reduction on these features using PCA to obtain sets of features in an $(d = 8)$-dimensional space.

## 4.4. Results

We show the main experimental results in Table 1. We see that SLoSH provides a consistent improvement on retrieval performance for all datasets when compared with non-distribution-based methods, especially when $L > d$. When compared with WE, SLoSH presents comparable retrieval performance on Point MNIST 2D and Oxford 5k

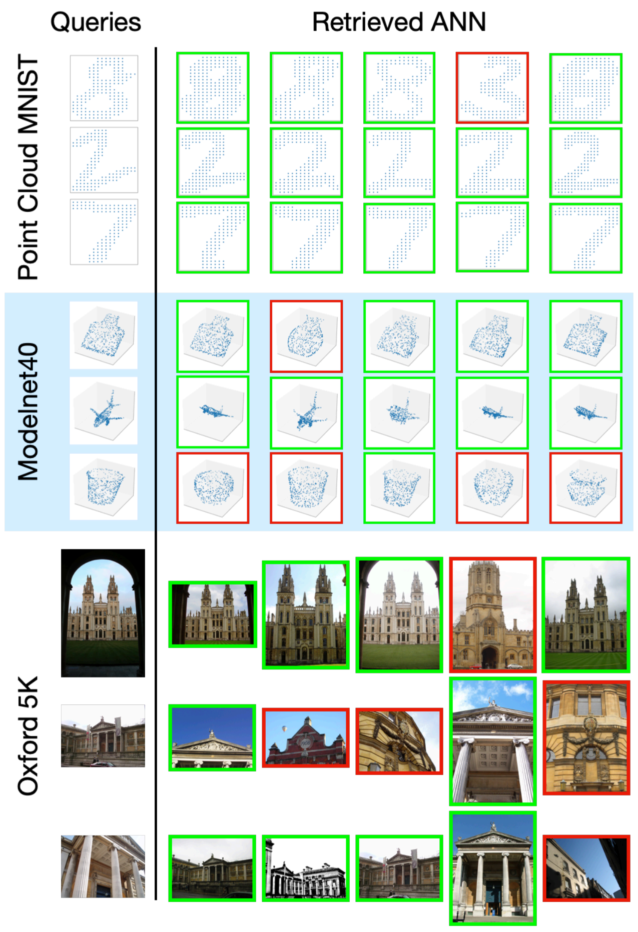

Figure 3. We provide a qualitative analysis of SLoSH. For each dataset, we show three sample queries and their retrieved samples, where green border highlights samples with the same class as the queries while red border denotes samples with a different class from the queries.

while providing a significant speed boost as shown in Figure 2, where we present an analysis of embedding time against retrieval accuracy for all methods to demonstrate their accuracy-efficiency trade-off. While WE provides strong performance, its high computational complexity may defeat the purpose of ANN algorithms. In Figure 3, we provide a qualitative analysis of sample queries and retrieved samples by SLoSH. On Point MNIST, SLoSH is able to retrieve with high precision; the only irrelevant sample is a digit 3 when the query is digit 8, and the two indeed resemble each other. For ModelNet40, we noticed that SLoSH is fairly good at retrieving samples for airplanes and bottles, but rather bad at retrieving samples for cups. We observe that for the cup query, SLoSH often retrieves irrelevant samples from vase, bowl and flowerpot. We argue that these classes are inherently harder to distinguish for two reasons: 1) they share similar attributes, and 2) their scales are not considered after normalization, which provide critical class information. Finally, for Oxford 5K, SLoSH also delivers decent retrieval performance. Since the quality of retrieval also depends on extracted convolution features, we believe SLoSH could benefit from a more powerful visual encoder.

## 4.5. Sensitivity Analysis

Next, we provide a sensitivity analysis of our approach with respect to three key hyperparameters of SLoSH: hash code length, number of slices, and reference sets.

**Sensitivity to code length.** For all datasets, we study the sensitivity of the different embeddings to the hashing code length used in LSH. We vary the code length from 16 to 1024, and report the average of Precision@k over five runs. Figure 4 shows the outcome of this study. We observe that as we increase the hash code length, SLoSH, gains consistent performance boosts, empirically validating the positive correlation between collision probability and code length shown in 3.4. The superiority of SLoSH compared to other baselines across different code lengths also suggests it has a higher lower bound ($P_1$) for similar samples.

**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 code lengths and report our results in Figure 5. Overall, we observe that increasing the number of slices brings better performance, which supports our theoretical results in 8. In addition, we observe that for low-dimensional set retrieval tasks such as Point MNIST (2-D) and ModelNet40 (3D), 16 slices are enough for SLoSH to reach the best performance. By contrast, on higher-dimension sets, such as Oxford 5k (8D), SLoSH requires a larger number of slices (128) before the performance saturates. This is anticipated as we mentioned in 3.2 that the upper bound of MC's approximation error is implicitly dependent on the set dimension.

**Sensitivity to the reference.** Finally, we study SLoSH's sensitivity to the choice of its reference set, $X_0$. We measure the performance of SLoSH on the three datasets and for various code lengths, when the reference set is formed by 1) using K-Means on the elements of all sets, 2) sampling the dataset, 3) sampling a uniform distribution, and 4) sampling a normal distribution. We find that on Point MNIST and ModelNet40, using a reference from KMeans or existing samples outperforms using a reference from random sampling, especially when the hash code length size is small, though we observe that the gap closes up when we increase the hash code length. By contrast, on Oxford 5k, we find that using reference from random sampling results in better performance. This suggests that the optimal reference set might be data-independent. To get the best performance from SLoSH, we recommend using a validation set to select the best reference set.

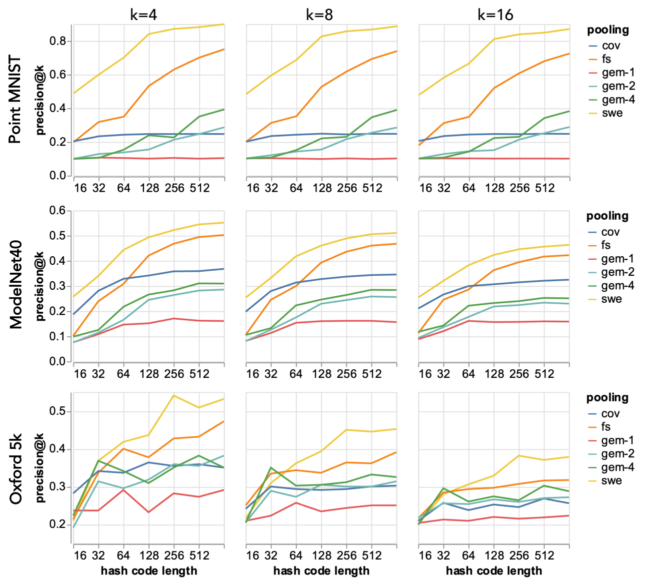

Figure 4. We show a sensitivity analysis of different methods regarding the hash code length of LSH. We notice that distribution-based pooling (SWE, FSPool) benefit more from longer hash code length than non-distribution-based pooling methods (GeM-1, GeM-2, GeM-4, Cov).

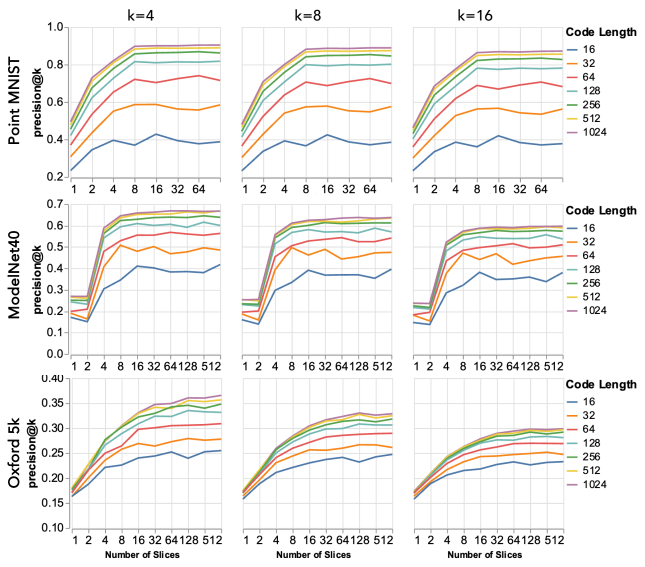

Figure 5. We provide a sensitivity analysis of SLoSH with respect to the number of slices used to construct SWE. Generally, we find increasing the number of slices improves retrieval performance, but there exists a saturation point after which more slices don't further boost performance.

## 5. 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 provide, for the first time, the probability of collision of LSH codes for sets as a function of the Sliced Wasserstein distance between the sets' corresponding empirical distributions. We then demonstrate the performance of SLoSH, in terms of retrieval accuracy as well as computational efficiency on three different set retrieval tasks: Point Cloud MNIST, ModelNet40, and the Oxford Buildings datasets. We demonstrate a significant retrieval accuracy boost over other data-independent baselines as well as a great boost in computational efficiency over the Wasserstein Embedding (WE). Finally, through various sensitivity studies, we demonstrate the impact of hash code length, number of slices, and choice of reference set on SLoSH's retrieval performance.

Despite its strong performance, we note that it might be currently challenging to apply SLoSH on higher-dimensional sets. Since SLoSH inherits the theoretical properties of SWE, we need a larger number of slices to get higher-quality embeddings when dealing with high-dimensional sets. We plan to propose methods to deal with this issue in future works.

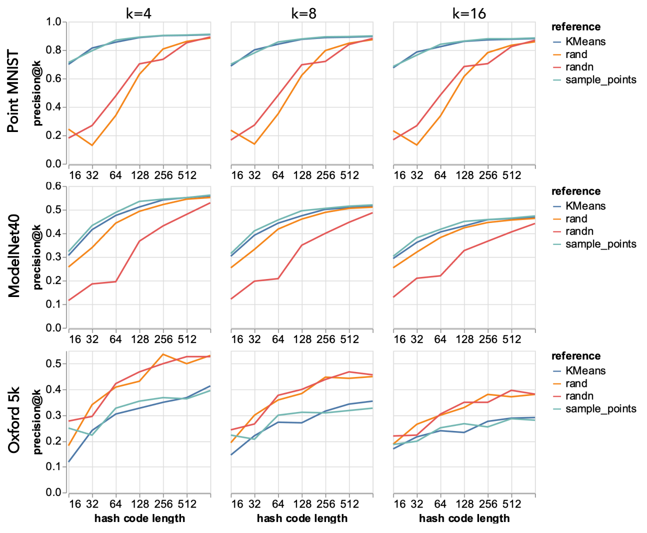

Figure 6. We present a sensitivity analysis on the choice of reference sets for SLoSH when using different code lengths. We find that SLoSH's sensitivity to the reference sets is related to hash code length, and there is not one reference function that achieves optimal performance on all datasets.

## 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. 5

[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 Nguyn, 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 Bonnotte. *Unidimensional and evolution methods for optimal transportation*. PhD thesis, Université Paris 11, France, 2013. 1

[13] Nicolas Courty, Rémi Flamary, and Mélanie Ducoffe. Learning wasserstein embeddings. *arXiv preprint arXiv:1710.07457*, 2017. 5

[14] 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

[15] 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

[16] 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

[17] Cristian Garcia. Point cloud mnist 2d, 2020. 5, 6

[18] 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

[19] 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

[20] 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

[21] Tony Jebara, Risi Kondor, and Andrew Howard. Probability product kernels. *The Journal of Machine Learning Research*, 5:819–844, 2004. 2

[22] 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

[23] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017. 6

[24] 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

[25] 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. 2

[26] Soheil Kolouri, Navid Naderializadeh, Gustavo K Rohde, and Heiko Hoffmann. Wasserstein embedding for graph learning. In *ICLR*, 2021. 2, 3, 5

[27] 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

[28] 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. 2, 3

[29] 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. 1, 2, 3

[30] Soheil Kolouri, Phillip E. Pope, Charles E. Martin, and Gustavo K. Rohde. Sliced Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2019. 3

[31] Soheil Kolouri, Akif B Tosun, John A Ozolek, and Gustavo K Rohde. A continuous linear optimal transport approach for pattern analysis in image datasets. *Pattern recognition*, 51:453–462, 2016. 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

[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. 5, 6

[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] Xinran Liu, Yikun Bai, Yuzhe Lu, Andrea Soltoggio, and Soheil Kolouri. Wasserstein task embedding for measuring task similarities. *arXiv preprint arXiv:2208.11726*, 2022. 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, 4

[39] 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

[40] 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. 3

[41] 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

[42] 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

[43] Navid Naderializadeh, Joseph F Comer, Reed Andrews, Heiko Hoffmann, and Soheil Kolouri. Pooling by sliced-wasserstein embedding. *Advances in Neural Information Processing Systems*, 34:3389–3400, 2021. 2, 3

[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. 3

[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. 2

[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. 3

[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. 1, 2

[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. 5, 6

[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

[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, 5

[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. 6

[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, Xuwang Yin, Abu Hasnat Mohammad Rubaiyat, Shiying Li, Soheil Kolouri, Akram Aldroubi, Jonathan M Nichols, and Gustavo K Rohde. Radon cumulative distribution transform subspace modeling for im-

age classification. *Journal of Mathematical Imaging and Vision*, 63:1185–1203, 2021. 3

[57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[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 Schoelkopf. 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. 1, 2

[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, 5

[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. 3, 5

[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. 5, 6

[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 Ravanbhakhsh, 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. 5

[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