Wasserstein Dictionary Learning: Optimal Transport-based unsupervised non-linear dictionary learning

Morgan A Schmitz, Matthieu Heitz, Nicolas Bonneel, Fred Maurice Ngolè Mboula, David Coeurjolly, Marco Cuturi, Gabriel Peyré, Jean-Luc Starck

Wasserstein Dictionary Learning: Optimal Transport-Based Unsupervised Nonlinear Dictionary Learning

Morgan A. Schmitz*, Matthieu Heitz†, Nicolas Bonneel†, Fred Ngolè‡, David Coeurjolly†, Marco Cuturi§, Gabriel Peyré¶, and Jean-Luc Starck*

Abstract. This paper introduces a new nonlinear dictionary learning method for histograms in the probability simplex. The method leverages optimal transport theory, in the sense that our aim is to reconstruct histograms using so-called displacement interpolations (a.k.a. Wasserstein barycenters) between dictionary atoms; such atoms are themselves synthetic histograms in the probability simplex. Our method simultaneously estimates such atoms, and, for each datapoint, the vector of weights that can optimally reconstruct it as an optimal transport barycenter of such atoms. Our method is computationally tractable thanks to the addition of an entropic regularization to the usual optimal transportation problem, leading to an approximation scheme that is efficient, parallel and simple to differentiate. Both atoms and weights are learned using a gradient-based descent method. Gradients are obtained by automatic differentiation of the generalized Sinkhorn iterations that yield barycenters with entropic smoothing. Because of its formulation relying on Wasserstein barycenters instead of the usual matrix product between dictionary and codes, our method allows for nonlinear relationships between atoms and the reconstruction of input data. We illustrate its application in several different image processing settings.

Key words. optimal transport, Wasserstein barycenter, dictionary learning

AMS subject classifications. 33F05, 49M99, 65D99, 90C08

1. Introduction.

The idea of dimensionality reduction is as old as data analysis [57]. Dictionary learning [41], independent component analysis [37], sparse coding [42], autoencoders [35] or most simply principal component analysis (PCA) are all variations of the idea that each datapoint of a high-dimensional dataset can be efficiently encoded as a low-dimensional vector. Dimensionality reduction typically exploits a sufficient amount of data to produce an encoding map of datapoints into smaller vectors, coupled with a decoding map able to reconstruct an approximation of the original datapoints using such vectors. Algorithms to carry out the encoding and/or the decoding can rely on simple linear combinations of vectors, as is the case with PCA and nonnegative matrix factorization. They can also be highly nonlinear and employ kernel methods [72] or neural networks for that purpose [35].

In this work, we consider a very specific type of encoding/decoding pair, which relies on optimal transport (OT) geometry between probability measures. OT geometry, also known as Wasserstein or earth mover's, defines a distance between two probability measures μ, ν by computing the minimal effort required to morph measure μ into measure ν. Monge's original interpretation [50] was that μ would stand for a heap of sand, which should be used to fill in a

*Astrophysics Department, IRFU, CEA, Université Paris-Saclay, F-91191 Gif-sur-Yvette, France and Université Paris-Diderot, AIM, Sorbonne Paris Cité, CEA, CNRS, F-91191 Gif-sur-Yvette, France (morgan.schmitz@cea.fr)
†Université de Lyon, CNRS/LIRIS, Lyon, France
‡LIST, Data Analysis Tools Laboratory, CEA Saclay, France
§Centre de Recherche en Economie et Statistique, Paris, France
¶DMA, ENS Ulm, Paris, France

hole in the ground of the shape of ν. The effort required to move the pile of sand is usually parameterized by a cost function to move one atom of sand from any location x in the support of μ to any location y in the support of ν (see Figure 1). Monge then considered the problem of finding the optimal (least costly) way to level the ground by transporting the heap into the hole. That cost defines a geometry between probability measures which has several attractive properties. In this paper we exploit the fact that shapes and, more generally, images can be cast as probability measures, and we propose several tools inherited from OT geometry, such as OT barycenters, to warp and average such images [77]. These tools can be exploited further to carry out non-linear inverse problems in a Wasserstein sense [14], and we propose in this work to extend this approach to carry out nonlinear dictionary learning on images using Wasserstein geometry.

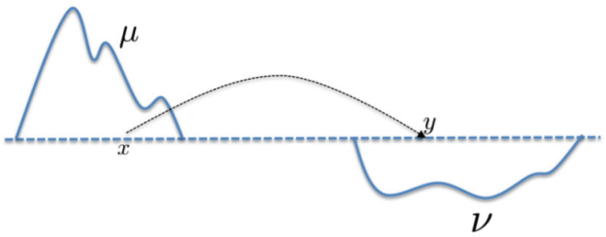


Figure 1: Graphical representation of the mass transportation problem. The minimal effort cost to transport one measure into the other defines the OT distance between μ and ν.

1.1. Previous works.

Linear dictionary learning. Several dimensionality reduction approaches rely on using a predefined orthogonal basis upon which datapoints can be projected. Such basis are usually defined without even looking at data, as is the case for Fourier transforms or wavelet-based dictionaries [47]. Dictionary learning methods instead underline the idea that dictionaries should be *customized* to fit a particular dataset in an optimal way. Suppose that the M datapoints of interest can be stored in a matrix $X = (x_1, \dots, x_M) \in \mathbb{R}^{N \times M}$. The aim of (linear) dictionary learning is to factorize the data matrix X using two matrices: a dictionary, D, whose elements (the atoms) have the same dimension N as those of X, and a list of codes Λ used to relate the two: $X \approx D\Lambda$.

When no constraints on D or Λ are given, and one simply seeks to minimize the Frobenius norm of the difference of X and $D\Lambda$, the problem amounts to computing the singular value decomposition of X or, equivalently, the diagonalization of the variance matrix of X. In practical situations, one may wish to enforce certain properties of that factorization, which can be done in practice by adding a prior or a constraint on the dictionary D, the codes Λ, or both. For instance, an l_0 or l_1 norm penalty on the codes yields a sparse representation of data [2, 46]. The sparsity constraint might instead be imposed upon the new components (or

atoms), as is the case for sparse PCA [21]. Properties other than sparsity might be desired, for example, statistical independence between the components, yielding independent component analysis (ICA [37]), or positivity of both the dictionary entries and the codes, yielding a nonnegative matrix factorization (NMF [41]). A third possible modification of the dictionary learning problem is to change the *fitting loss* function that measures the discrepancy between a datapoint and its reconstruction. When data lies in the nonnegative orthant, Lee and Seung have shown, for instance, the interest of considering the Kullback-Leibler (KL) divergence to compute such a loss [41] or, more recently, the Wasserstein distance [65], as detailed later in this section. More advanced fitting losses can also be derived using probabilistic graphical models, such as those considered in the topic modeling literature [12].

Nonlinear dictionary learning. The methods described above are linear in the sense that they attempt to reconstruct each datapoint x_i by a linear combination of a few dictionary elements. Nonlinear dictionary learning techniques involve reconstructing such datapoints using nonlinear operations instead. Autoencoders [35] propose using neural networks and to use their versatility to encode datapoints into low-dimensional vectors and later decode them with another network to form a reconstruction. The main motivation behind principal geodesic analysis [24] is to build such nonlinear operations using geometry, namely by replacing linear interpolations with geodesic interpolations. Of particular relevance to our paper is the body of work that relies on Wasserstein geometry to compute geodesic components [11, 13, 74, 85] (see subsection 5.1).

More generally, when data lies on a Riemannian manifold for which Riemannian exponential and logarithmic maps are known, Ho, Xie and Vemuri propose a generalization of both sparse coding and dictionary learning [36]. Nonlinear dictionary learning can also be performed by relying on the "kernel trick", which allows one to learn dictionary atoms that lie in some feature space of higher, or even infinite, dimension [32, 45, 82]. Equiangular kernel dictionary learning, proposed by Quan, Bao, and Ji, further enforces stability of the learned sparse codes [62]. Several problems where data is known to belong to a specific manifold are well studied within this framework, *e.g.* sparse coding and dictionary learning for Grassmann manifolds [33], or for positive definite matrices [34], and methods to find appropriate kernels and make full use of the associated manifold's geometry have been proposed for the latter [44]. Kernel dictionary learning has also been studied for the (nonlinear) adaptive filtering framework, where Gao et al. propose an online approach that discards obsolete dictionary elements as new inputs are acquired [28]. These methods rely on the choice of a particular feature space and an associated kernel and achieve nonlinearity through the use of the latter. The learned dictionary atoms then lie in that feature space. Conversely, our proposed approach requires no choice of kernel. Moreover, the training data and the atoms we learn belong to the same probability simplex, which allows for easy representation and interpretation; *e.g.* our learned atoms can (depending on the chosen fitting loss) capture the extreme states of a transformation undergone by the data. This is opposed to kernel dictionary atoms, which cannot be naturally represented in the same space as datapoints because of their belonging to the chosen high-dimensional feature space.

Computational optimal transport. Optimal transport has seen significant interest from mathematicians in recent decades [64, 79, 83]. For many years, that theory was, however, of

limited practical use and mostly restricted to the comparison of small histograms or point clouds, since typical algorithms used to compute them, such as the auction algorithm [10] or the Hungarian algorithm [39], were intractable beyond a few hundred bins or points. Recent approaches [63, 75] have ignited interest for fast yet faithful approximations of OT distances. Of particular interest to this work is the entropic regularization scheme proposed by Cuturi [18], which finds its roots in the gravity model used in transportation theory [23]. This regularization can also be tied to the relation between OT and Schrödinger's problem [73] (as explored by Léonard [43]). Whereas the original OT problem is a *linear* problem, regularizing it with an entropic regularization term results in a strictly convex problem with a unique solution which can be solved with Sinkhorn's fixed-point algorithm [76], a.k.a. block coordinate ascent in the dual regularized OT problem. That iterative fixed-point scheme yields a numerical approach relying only on elementwise operations on vectors and matrix-vector products. The latter can in many cases be replaced by a separable convolution operator [77], forgoing the need to manipulate a full cost matrix of prohibitive dimensions in some use cases of interest (*e.g.* when input measures are large images).

Wasserstein barycenters. Agueh and Carlier introduced the idea of a Wasserstein barycenter in the space of probability measures [1], namely Fréchet means [26] computed with the Wasserstein metric. Such barycenters are the basic building block of our proposal of a nonlinear dictionary learning scheme with Wasserstein geometry. Agueh and Carlier studied several properties of Wasserstein barycenters and showed very importantly that their exact computation for empirical measures involves solving a multimarginal optimal transport problem, namely a linear program with the size growing exponentially with the size of the support of the considered measures.

Since that work, several algorithms have been proposed to efficiently compute these barycenters [15, 16, 63, 78, 86]. The computation of such barycenters using regularized distances [19] is of particular interest to this work. Cuturi and Peyré [20] use entropic regularization and duality to cast a wide range of problems involving Wasserstein distances (including the computation of Wasserstein barycenters) as simple convex programs with closed form derivatives. They also illustrate the fact that the smoothness introduced by the addition of the entropic penalty can be *desirable*, beyond its computational gains, in the case of the Wasserstein barycenter problem. Indeed, when the discretization grid is small, its true optimum can be highly unstable, which is counteracted by the smoothing introduced by the entropy [20, §3.4]. The idea of performing iterative Bregman projections to compute approximate Wasserstein distances can be extended to the barycenter problem, allowing its direct computation using a generalized form of the Sinkhorn algorithm [8]. Chizat et al. recently proposed a unifying framework for solving unbalanced optimal transport problems [17], including computing a generalization of the Wasserstein barycenter.

Wasserstein barycentric coordinates. An approach to solving the inverse problem associated with Wasserstein barycenters was recently proposed [14]: Given a database of S histograms, a vector of S weights can be associated to any new input histogram, such that the barycenter of that database with those weights approximates as closely as possible the input histogram. These weights are obtained by automatic differentiation (with respect to the weights) of the generalized Sinkhorn algorithm that outputs the approximate Wasserstein

barycenter. This step can be seen as an analogy of, given a dictionary D, finding the best vector of weights Λ that can help reconstruct a new datapoint using the atoms in the dictionary. That work can be seen as a precursor for our proposal, whose aim is to learn both weights *and* dictionary atoms.

Applications to image processing. OT was introduced into the computer graphics community by Rubner, Tomasi, and Guibas [67] to retrieve images from their color distribution, by considering images as distributions of pixels within a 3-dimensional color space. Color processing has remained a recurring application of OT, for instance to color grade an input image using a photograph of a desired color style [60], or using a database of photographs [14], or to harmonize multiple images' colors [15]. Another approach considers grayscale images as 2-dimensional histograms. OT then allows one to find a transport-based warping between images [31, 49]. Further image processing applications are reviewed in the habilitation dissertation of Papadakis [56].

Wasserstein loss and fidelity. Several recent papers have investigated the use of OT distances as fitting losses that have desirable properties that KL or Euclidean distances cannot offer. We have already mentioned generalizations of PCA to the set of probability measures via the use of OT distances [11, 74]. Sandler and Lindenbaum first considered the NMF problem with a Wasserstein loss [69]. Their computational approach was, however, of limited practical use. More scalable algorithms for Wasserstein NMF and (linear) dictionary learning were subsequently proposed [65]. The Wasserstein distance was also used as a loss function with desirable robustness properties to address multilabel supervised learning problems [27].

Using the Wasserstein distance to quantify the fit between data (an empirical measure) and a parametric family of densities, or a generative model defined using a parameterized push-forward map of a base measure, has also received ample attention in the recent literature. Theoretical properties of such estimators were established by Bassetti, Bodini, and Regazzini [6] and Bassetti and Regazzini [7], and additional results by Bernton et al. [9]. Entropic smoothing has facilitated the translation of these ideas into practical algorithms, as illustrated in the work by Montavon, Müller, and Cuturi, who proposed to estimate the parameters of restricted Boltzmann machines using the Wasserstein distance instead of the KL divergence [51]. Purely generative models, namely, degenerate probability measures defined as the push-forward of a measure supported on a low-dimensional space into a high-dimensional space using a parameterized function, have also been fitted to observations using a Wasserstein loss [9], allowing for density fitting without having to choose summary statistics (as is often the case with usual methods). The Wasserstein distance has also been used in the context of generative adversarial networks (GANs) [5]. In that work, the authors use a proxy to approximate the 1-Wasserstein distance. Instead of computing the 1-Wasserstein distance using 1-Lipschitz functions, a classic result from Kantorovich's dual formulation of OT (see Theorem 1.14 in Villani's book [83]), the authors restrict that set to multilayer networks with rectified linear units and boundedness constraints on weights, which allows them to enforce some form of Lipschitzness of their networks. Unlike the entropic smoothing used in this paper, that approximation requires solving a nonconvex problem whose optimum, even if attained, would be arbitrarily far from the true Wassertein distance. More recently, Genevay, Peyré, and Cuturi introduced a general scheme for using OT distances as the loss in generative models [29], which

relies on both the entropic penalty and automatic differentiation of the Sinkhorn algorithm. Our work shares some similarities with that paper, since we also propose automatically differentiating the Sinkhorn iterations used in Wasserstein barycenter computations.

1.2. Contributions. In this paper, we introduce a new method for carrying out nonlinear dictionary learning for probability histograms using OT geometry. Nonlinearity comes from the fact that we replace the usual linear combination of dictionary atoms by Wasserstein barycenters. Our goal is to reconstruct datapoints using the closest (according to any arbitrary fitting loss on the simplex, not necessarily the Wasserstein distance) Wasserstein barycenter to that point using the dictionary atoms. Namely, instead of considering linear reconstructions for $X \approx D\Lambda$, our aim is to approximate columns of $X \approx \mathbf{P}(D, \Lambda)$ using the $\mathbf{P}$ operator which maps atoms D with lists of weights Λ to their respective barycenters.

Similar to many traditional dictionary learning approaches, this is achieved by finding local minima of a nonconvex energy function. To do so, we propose using automatic differentiation of the iterative scheme used to compute Wasserstein barycenters. We can thus obtain gradients with respect to both the dictionary atoms and the weights that can then be used within one's solver of choice (in this work, we chose to use an off-the-shelf quasi-Newton approach and perform both dictionary and code updates simultaneously).

Our nonlinear dictionary learning approach makes full use of the Wasserstein space's properties, as illustrated in Figure 2: two atoms are learned from a dataset made up of five discretized Gaussian distributions in 1D, each slightly translated on the grid. Despite the simplicity of the transformation (translation), linear generative models fail to capture the changes of the geometrical space, as opposed to our OT approach. Moreover, the atoms we learn are also discrete measures, unlike the PCA and NMF components.

We also offer several variants and improvements to our method:

- Arbitrarily sharp reconstructions can be reached by performing the barycenter computation in the log-domain;
- We offer a general method to make use of the separability of the kernel involved and greatly alleviate the computational cost of this log-domain stabilization;
- Our representation is learned from the differentiation of an iterative, Sinkhorn-like algorithm, whose convergence can be accelerated by using information from previous Sinkhorn loops at each initialization (warm start), or adding a momentum term to the Sinkhorn iterations (heavyball);
- We expand our method to the unbalanced transport framework.

Part of this work was previously presented as a conference proceedings [70], featuring an initial version of our method, without any of the above improvements and variants, and in the case where we were only interested in learning two different atoms.

Additional background on OT is given in section 2. The method itself and an efficient implementation are presented in section 3. We highlight other extensions in section 4. We showcase its use in several image processing applications in section 5.

1.3. Notations. We denote Σ_d the simplex of $\mathbb{R}^d$, that is,

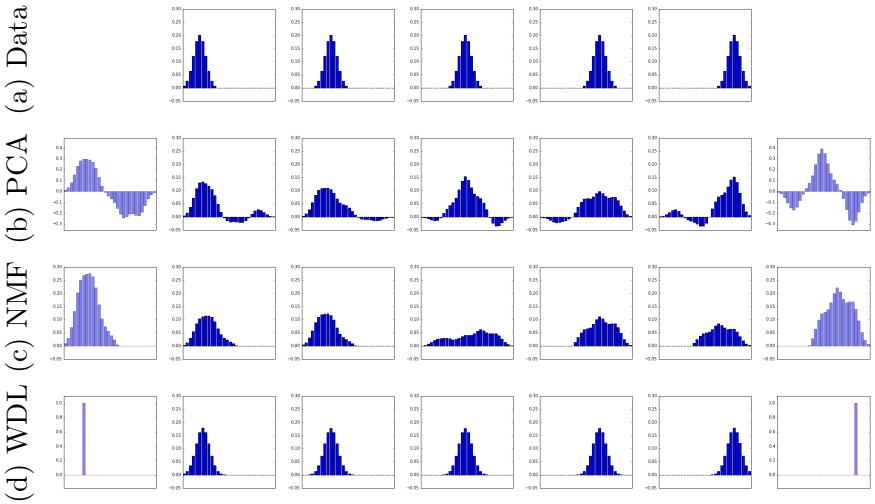


Figure 2: Top row: data points. Bottom three rows: On the far sides, in purple, are the two atoms learned by PCA, NMF and our method (WDL), respectively. In between the two atoms are the reconstructions of the five datapoints for each method. The latter two were relaunched a few times with randomized initializations and the best local minimum was kept. As discussed in section 2, the addition of an entropy penalty to the usual OT program causes a blur in the reconstructions. When the parameter associated with the entropy is high, our method yields atoms that are sharper than the dataset on which it was trained, as is observed here where the atoms are Dirac despite the dataset consisting of discretized Gaussians. See subsection 4.1 for a method to reach arbitrarily low values of the entropy parameter and counteract the blurring effect.

$$\Sigma_d := \left\{ u \in \mathbb{R}_+^d, \sum_{i=1}^d u_i = 1 \right\}.$$

For any positive matrix T, we define its negative entropy as

$$H(T) := \sum_{i,j} T_{ij} \log(T_{ij} - 1).$$

$\odot$ denotes the Hadamard product between matrices or vectors. Throughout this paper, when applied to matrices, $\prod, \div$, and exp notations refer to elementwise operators. The scalar product between two matrices denotes the usual inner product, that is,

$$\langle A, B \rangle := \mathrm{Tr}(A^\top B) = \sum_{i,j} A_{ij} B_{ij},$$

where $A^\top$ is the transpose of A. For $(p, q) \in \Sigma_N^2$, we denote their set of couplings as

$$\Pi(p, q) := \left\{ T \in \mathbb{R}_+^{N\times N}, T\mathbb{1}_N = p, T^\top \mathbb{1}_N = q \right\}, \tag{1}$$

where $\mathbb{1}_N = (1, \dots, 1)^\top \in \mathbb{R}^N$. Δ denotes the diag operator, such that if $u \in \mathbb{R}^N$, then

$$\Delta(u) := \begin{pmatrix} u_1 & & \\ & \ddots & \\ & & u_N \end{pmatrix} \in \mathbb{R}^{N\times N}.$$

ι is the indicator function, such that for two vectors u, v,

$$\iota_{\{u\}}(v) = \begin{cases} 0 & \text{if } u = v, \\ +\infty & \text{otherwise,} \end{cases} \tag{2}$$

and KL(.|.) is their KL divergence, defined here as

$$\mathrm{KL}(u|v) = \sum_i u_i \log\left(\frac{u_i}{v_i}\right) - u_i + v_i.$$

For a concatenated family of vectors $t = \left[t_1^\top, \dots, t_S^\top\right]^\top \in \mathbb{R}^{NS}$, we write the ith element of t_s as $[t_s]_i$. We denote the rows of matrix M as $M_{i\cdot}$ and its columns as $M_{\cdot j}$. I_N and $\mathbf{0}_{N\times N}$ are the $N \times N$ identity and zero matrices, respectively.

2. Optimal transport.

2.1. OT distances. In the present work, we restrict ourselves to the discrete setting, *i.e.* our measures of interest will be histograms, discretized on a fixed grid of size N (Eulerian discretization), and represented as vectors in Σ_N. In this case, the cost function is represented as a cost matrix $C \in \mathbb{R}^{N\times N}$, containing the costs of transportation between any two locations in the discretization grid. The OT distance between two histograms $(p, q) \in \Sigma_N^2$ is the solution to the discretized Monge–Kantorovich problem:

$$W(p, q) := \min_{T\in\Pi(p,q)} \langle T, C\rangle.$$

As defined in (1), $\Pi(p, q)$ is the set of admissible couplings between p and q, that is, the set of matrices with rows summing to p and columns to q. A solution, $T^* \in \mathbb{R}^{N\times N}$, is an optimal transport plan.

Villani's books give extended theoretical overviews of OT [83, 84] and, in particular, several properties of such distances. The particular case where the cost matrix is derived from a metric on the chosen discretization grid yields the so-called Wasserstein distance (sometimes called the earth mover's distance). For example, if $C_{ij} = \|x_i - x_j\|_2^2$ (where x_i, x_j are the positions on the grid), the above formulation yields the squared 2-Wasserstein distance, the square-root

of which is indeed a distance in the mathematical sense. Despite its intuitive formulation, the computation of Wasserstein distances grows supercubicly in N, making them impractical as dimensions reach the order of one thousand grid points. This issue has motivated the recent introduction of several approximations that can be obtained at a lower computational cost (see Computational optimal transport, subsection 1.1). Among such approximations, the entropic regularization of OT distances [18] relies on the addition of a penalty term as follows:

$$W_\gamma(p,q) := \min_{T\in\Pi(p,q)} \langle T, C\rangle + \gamma H(T), \tag{3}$$

where $\gamma > 0$ is a hyperparameter. As $\gamma \to 0$, W_γ converges to the original Wasserstein distance, while higher values of γ promote more diffuse transport matrices. The addition of a negentropic penalty makes the problem γ-strongly convex; first-order conditions show that the problem can be analyzed as a matrix-scaling problem which can be solved using Sinkhorn's algorithm [76] (also known as the iterative proportional fitting procedure (IPFP) [22]). The Sinkhorn algorithm can be interpreted in several ways: for instance, it can be thought of as an alternate projection scheme under a KL divergence for couplings [8] or as a block-coordinate ascent on a dual problem [19]. The Sinkhorn algorithm consists in using the following iterations for $l \geq 1$, starting with $b^{(0)} = \mathbb{1}_N$:

$$\begin{aligned} a^{(l)} &= \frac{q}{K^\top b^{(l-1)}}, \\ b^{(l)} &= \frac{p}{K a^{(l)}}, \end{aligned} \tag{4}$$

where $K := \exp(\frac{-C}{\gamma})$ is the elementwise exponential of the negative of the rescaled cost matrix. Note that when γ gets close to 0, some values of K become negligible, and values within the scaling vectors, $a^{(l)}$ and $b^{(l)}$, can also result in numerical instability in practice (we will study workarounds for that issue in subsection 4.1). Application of the matrix K can often be closely approximated by a separable operation [77] (see subsubsection 4.1.2 for separability even in the log-domain). In the case where the histograms are defined on a uniform grid and the cost matrix is the squared Euclidean distance, the convolution kernel is simply Gaussian with standard deviation $\sqrt{\gamma/2}$. The two vectors $a^{(l)}, b^{(l)}$ converge linearly towards the optimal scalings [25] corresponding to the optimal solution of (3). Notice finally that the Sinkhorn algorithm at each iteration $l \geq 1$ results in an approximate optimal transport matrix $T^{(l)} = \Delta(b^{(l)})K\Delta(a^{(l)})$.

2.2. Wasserstein barycenter. Analogous to the usual Euclidean barycenter, the Wasserstein barycenter of a family of measures is defined as the minimizer of the (weighted) sum of squared Wasserstein distances from the variable to each of the measures in that family [1]. For measures with the same discrete support, we define, using entropic regularization, the barycenter of histograms $(d_1, \ldots, d_S) \in (\Sigma_N)^S$ with barycentric weights $\lambda = (\lambda_1, \ldots, \lambda_S) \in \Sigma_S$ as

$$P\,(D,\lambda) := \operatorname*{argmin}_{u\in\Sigma_N} \sum_{s=1}^{S} \lambda_s W_\gamma(d_s, u), \tag{5}$$

where $D := (d_1^\top, \dots, d_S^\top)^\top \in \mathbb{R}^{NS}$. The addition of the entropy term ensures strict convexity and thus that the Wasserstein barycenter is uniquely defined. It also yields a simple and efficient iterative scheme to compute approximate Wasserstein barycenters, which can be seen as a particular case of the unbalanced OT setting [17]. This scheme, a generalization of the Sinkhorn algorithm, once again relies on two scaling vectors:

$$a_s^{(l)} = \frac{d_s}{K b_s^{(l-1)}}, \tag{6}$$

$$P^{(l)}(D, \lambda) = \prod_{s=1}^{S} \left(K^\top a_s^{(l)} \right)^{\lambda_s}, \tag{7}$$

$$b_s^{(l)} = \frac{P^{(l)}(D, \lambda)}{K^\top a_s^{(l)}}, \tag{8}$$

where, as before, $K = \exp(\frac{-C}{\gamma})$. In this case, however, the scaling vectors are of size NS, such that $a^{(l)} = \left(a_1^{(l)\top}, \dots, a_S^{(l)\top} \right)^\top, b^{(l)} = \left(b_1^{(l)\top}, \dots, b_S^{(l)\top} \right)^\top$ and $b^{(0)} = \mathbb{1}_{NS}$. Note that one can perform both scaling vector updates at once (and avoid storing both) by plugging one of (6), (8) into the other. An illustration of the Wasserstein barycenter, as well as the impact of the γ parameter, is given in Figure 3.

3. Wasserstein dictionary learning.

3.1. Overview. Given data $X \in \mathbb{R}^{N \times M}$ in the form of histograms, *i.e.*, each column $x_i \in \Sigma_N$ (for instance a list of M images with normalized pixel intensities), and the desired number of atoms S, we aim to learn a dictionary D made up of histograms $(d_1, \dots, d_S) \in (\Sigma_N)^S$ and a list of barycentric weights $\Lambda = (\lambda_1, \dots, \lambda_M) \in (\Sigma_S)^M$ so that for each input, $P(D, \lambda_i)$ is the best approximation of x_i according to some criterion $\mathcal{L}$ (see Table 1 for examples). Namely, our representation is obtained by solving the problem

$$\min_{D,\Lambda} \mathcal{E}(D, \Lambda) := \sum_{i=1}^{M} \mathcal{L}\left(P(D, \lambda_i), x_i\right). \tag{9}$$

Note the similarity between the usual dictionary learning formulation (see Linear dictionary learning, subsection 1.1) and the one above. In our case, however, the reconstruction of the original data happens *via* the nonlinear Wasserstein barycenter operator, $\mathbf{P}(D, \Lambda) = (P(D, \lambda_i))_i$, instead of the (linear) matrix product $D\Lambda$.

Differentiation of (9) relies in part on the computation of the Wasserstein barycenter operator's Jacobians with regard to either the barycentric weights or the atoms. While it is possible to obtain their analytical formulae, for example by using the fact that Sinkhorn updates (7)-(8) become fixed-point equations when convergence is reached, they rely on solving a linear system of prohibitive dimensionality for our settings of interest where N is typically large (Bonneel, Peyré, and Cuturi derived the expression with regard to barycentric weights and discussed the issue in [14, §4.1]). Moreover, in practice, the true Wasserstein barycenters

(a) Wasserstein simplex; $\gamma = 8$ (b) Wasserstein simplex; $\gamma = 1$

Figure 3: Wasserstein simplices: barycenters of the three images in the corners with varying barycentric weights. Middle row: $\lambda = \left[\frac{1}{2}, \frac{1}{2}, 0\right], \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right], \left[0, \frac{1}{2}, \frac{1}{2}\right]$. Bottom row, center: $\left[\frac{1}{2}, 0, \frac{1}{2}\right]$.

with entropic penalty $P(D, \lambda_i)$ are unknown and approximated by sufficient Sinkhorn iterations (7)–(8). As is now common practice in some machine learning methods (a typical example being backward propagation for neural nets), and following recent works [14], we instead take an approach in the vein of automatic differentiation [30]. That is, we recursively differentiate the iterative scheme yielding our algorithm instead of the analytical formula of our Wasserstein barycenter. In our case, this is the generalization of the Sinkhorn algorithm for barycenters. Instead of (9), we thus aim to minimize

$$\min_{D,\Lambda} \mathcal{E}_L(D, \Lambda) := \sum_{i=1}^{M} \mathcal{L}\left(P^{(L)}(D, \lambda_i), x_i\right), \tag{10}$$

where $P^{(L)}$ is the approximate barycenter after L iterations, defined as in (7). Even when using an entropy penalty term, we have no guarantee on the convexity of the above problem, whether jointly in D and Λ or for each separately, contrary to the case of OT distance computation in (3). We thus aim to reach a local minimum of energy landscape $\mathcal{E}_L$ by computing its gradients and applying a descent method. By additivity of $\mathcal{E}_L$ and without loss of generality, we will focus on the derivations of such gradients for a single datapoint $x \in \Sigma_N$ (in which case Λ only comprises one list of weights $\lambda \in \Sigma_S$). Differentiation of (10) yields

$$\nabla_D \mathcal{E}_L(D, \Lambda) = \left[\partial_D P^{(L)}(D, \lambda)\right]^\top \nabla \mathcal{L}(P^{(L)}(D, \lambda), x), \tag{11}$$

$$\nabla_\lambda \mathcal{E}_L(D, \Lambda) = \left[\partial_\lambda P^{(L)}(D, \lambda)\right]^\top \nabla \mathcal{L}(P^{(L)}(D, \lambda), x). \tag{12}$$

The right-hand term in both cases is the gradient of the loss which is typically readily computable (see Table 1) and depends on the choice of fitting loss. The left-hand terms are the Jacobians of the Wasserstein barycenter operator with regard to either the weights or the dictionary. These can be obtained either by performing the analytical differentiation of

the $P^{(l)}$ operator, as is done in subsection 3.2 (and Appendix A), or by using an automatic differentiation library such as Theano [80]. The latter approach ensures that the complexity of the backward loop is the same as that of the forward, but it can lead to memory problems due to the storing of all objects being part of the gradient computation graph (as can be the case, for instance, when performing the forward Sinkhorn loop in the log-domain as in subsubsection 4.1.1; for this specific case, an alternative is given in subsubsection 4.1.2). The resulting numerical scheme relies only on elementwise operations and on the application of the matrix K (or its transpose) which often amounts to applying a separable convolution [77] (see subsubsection 4.1.2). The resulting algorithm is given in Algorithm 1. At first, a "forward" loop is performed, which amounts to the exact same operations as those used to compute the approximate Wasserstein barycenter using updates (7)-(8) (the barycenter for current weights and atoms is thus computed as a by-product). Two additional vectors of size SNL are stored and then used in the recursive backward differentiation loops that compute the gradients with regard to the dictionary and the weights.

Name	$\mathcal{L}(p,q)$	$\nabla\mathcal{L}$
Total variation	$\|p-q\|_1$	$\operatorname{sign}(p-q)$
Quadratic	$\|p-q\|_2^2$	$2(p-q)$
Kullback-Leibler	$\mathrm{KL}(p\|q)$	$\log(p/q)-1$
Wasserstein[1]	$W_\gamma^{(L)}(p,q)$	$\gamma\log(a^{(L)})$

Table 1: Examples of similarity criteria and their gradient in p. See Figure 14 for the atoms yielded by our method for these various fitting losses.

Using the above scheme to compute gradients, or its automatically computed counterpart from an automatic differentiation tool, one can find a local minimum of the energy landscape (10), and thus the eventual representation Λ and dictionary D, by applying any appropriate optimization method under the constraints that both the atoms and the weights belong to their respective simplices Σ_N, Σ_S.

For the applications shown in section 5, we chose to enforce these constraints through the following change of variables

$$\forall i, d_i := F_N(\alpha_i) := \frac{\mathrm{e}^{\alpha_i}}{\sum_{j=1}^N \mathrm{e}^{[\alpha_i]_j}}, \qquad \lambda := F_S(\beta) := \frac{\mathrm{e}^{\beta}}{\sum_{j=1}^S \mathrm{e}^{\beta_j}}.$$

The energy to minimize (with regard to α, β) then reads as

$$\mathcal{G}_L(\alpha,\beta) := \mathcal{E}_L(F(\alpha), F_S(\beta)), \tag{13}$$

where $F(\alpha) := (F_N(\alpha_1), \dots, F_N(\alpha_S)) = D$. Differentiating (13) yields

[1] In this case, the loss is computed iteratively as explained in subsection 2.1, and $a^{(L)}$ in the gradient's expression is obtained after L iterations as in (4).

Algorithm 1 `SinkhornGrads`: Computation of dictionary and barycentric weights gradients

Inputs: Data $x \in \Sigma_N$, atoms $d_1, \dots, d_S \in \Sigma_N$, current weights $\lambda \in \Sigma_S$

comment: Sinkhorn loop

$\forall s, b_s^{(0)} := \mathbf{1}_N$

for $l = 1$ to L step 1 do

$\quad \forall s, \varphi_s^{(l)} := K^\top \frac{d_s}{K b_s^{(l-1)}}$

$\quad p := \prod_s \left(\varphi_s^{(l)}\right)^{\lambda_s}$

$\quad \forall s, b_s^{(l)} := \frac{p}{\varphi_s^{(l)}}$

od

comment: Backward loop - weights

$w := \mathbf{0}_S$

$r := \mathbf{0}_{S \times N}$

$g := \nabla \mathcal{L}(p, x) \odot p$

for $l = L$ to 1 step -1 do

$\quad \forall s, w_s := w_s + \langle \log \varphi_s^{(l)}, g \rangle$

$\quad \forall s, r_s := -K^\top \left(K \left(\frac{\lambda_s g - r_s}{\varphi_s^{(l)}} \right) \odot \frac{d_s}{(K b_s^{(l-1)})^2} \right) \odot b_s^{(l-1)}$

$\quad g := \sum_s r_s$

od

comment: Backward loop - dictionary

$y := \mathbf{0}_{S \times N}$

$z := \mathbf{0}_{S \times N}$

$n := \nabla \mathcal{L}(p, x)$

for $l = L$ to 1 step -1 do

$\quad \forall s, c_s := K((\lambda_s n - z_s) \odot b_s^{(l)})$

$\quad \forall s, y_s := y_s + \frac{c_s}{K b_s^{(l-1)}}$

$\quad \forall s, z_s := -\frac{\mathbb{1}_N}{\varphi_s^{(l-1)}} \odot K^\top \frac{d_s \odot c_s}{(K b_s^{(l-1)})^2}$

$\quad n := \sum_s z_s$

od

Outputs: $P^{(L)}(D, \lambda) := p, \nabla_D \mathcal{E}^{(L)} := y, \nabla_\lambda \mathcal{E}^{(L)} := w$

$$\nabla_\alpha \mathcal{G}_L(\alpha, \beta) = [\partial F(\alpha)]^\top \nabla_D \mathcal{E}_L\left(F(\alpha), F_S(\beta)\right) = [\partial F(\alpha)]^\top \nabla_D \mathcal{E}_L\left(D, \Lambda\right),$$
$$\nabla_\beta \mathcal{G}_L(\alpha, \beta) = [\partial F_S(\beta)]^\top \nabla_\lambda \mathcal{E}_L\left(F(\alpha), F_S(\beta)\right) = [\partial F_S(\beta)]^\top \nabla_\lambda \mathcal{E}_L\left(D, \Lambda\right),$$

where $[\partial F_p(u)]^\top = \partial F_p(u) = \left(I_p - F_p(u)\mathbb{1}_p^\top\right) \Delta\left(F_p(u)\right)$, p being either N or S for each atom or the weights, respectively, and both derivatives of $\mathcal{E}_L$ are computed using either automatic differentiation or as given in (11), (12) with Algorithm 1 (see subsection 3.2). The optimization can then be performed with no constraints over α, β.

Since the resulting problem is one where the function to minimize is differentiable and we are left with no constraints, in this work we chose to use a quasi-Newton method (though our approach can be used with any appropriate solver); that is, at each iteration t, an approximation of the inverse Hessian matrix of the objective function, $B^{(t)}$, is updated, and the logistic variables for the atoms and weights are updated as

$$\alpha^{(t+1)} := \alpha^{(t)} - \rho_\alpha^{(t)} B_\alpha^{(t)} \nabla_\alpha \mathcal{G}_L(\alpha, \beta), \qquad \beta^{(t+1)} := \beta^{(t)} - \rho_\beta^{(t)} B_\beta^{(t)} \nabla_\beta \mathcal{G}_L(\alpha, \beta),$$

where the $\rho^{(t)}$ are step sizes. An overall algorithm yielding our representation in this particular setup of quasi-Newton after a logistic change of variables is given in Algorithm 2.

In the applications of section 5, $B^{(t)}$ and $\rho^{(t)}$ were chosen using an off-the-shelf L-BFGS solver [52]. We chose to perform updates to atoms and weights simultaneously. Note that in this case, both are fed to the solver of choice as a concatenated vector. It is then beneficial to add a "variable scale" hyperparameter ζ and to multiply all gradient entries related to the weights by that value. Otherwise, the solver might reach its convergence criterion when approaching a local minimum with regards to either dictionary atoms or weights, even if convergence is not yet achieved in the other. Setting either a low or high value of ζ bypasses the problem by forcing the solver to keep optimizing with regard to one of the two variables in particular. In practice, and as expected, we have observed that relaunching the optimization with different ζ values upon convergence can increase the quality of the learned representation. While analogous to the usual alternated optimization scheme often used in dictionary learning problems, this approach avoids having to compute two different forward Sinkhorn loops to obtain the derivatives in both variables.

Algorithm 2 Quasi-Newton implementation of the Wasserstein dictionary learning algorithm

Inputs: Data $X \in \mathbb{R}^{N \times M}$, initial guesses $\alpha^{(0)}, \beta_1^{(0)}, \dots, \beta_M^{(0)}$, convergence criterion
$t := 0$
while convergence not achieved do
  $D^{(t)} := F(\alpha^{(t)})$
  $\alpha^{(t+1)} := \alpha^{(t)}$
  for $i = 1$ to M step 1 do
    $\lambda_i^{(t)} := F_S(\beta_i^{(t)})$
    $p_i, g_i^D, g_i^\lambda :=$ `SinkhornGrads`$(x_i, D^{(t)}, \lambda_i^{(t)})$
    Select $\rho_\alpha^{(t)}, \rho_i^{(t)} B_\alpha^{(t)}, B_i^{(t)}$ (`L-BFGS`)
    $\alpha^{(t+1)} := \alpha^{(t+1)} - \rho_\alpha^{(t)} B_\alpha^{(t)} \partial F(\alpha^{(t)}) g_i^D$
    $\beta_i^{(t+1)} := \beta_i^{(t)} - \rho_i^{(t)} B_i^{(t)} \partial F_S(\beta_i^{(t)}) g_i^\lambda$
  od
  $t := t + 1$
od
Outputs: $D = F\left(\alpha^{(t)}\right)$, $\Lambda = \left(F_S\left(\beta_1^{(t)}\right), \dots, F_S\left(\beta_S^{(t)}\right)\right)$

3.2. Backward recursive differentiation. To differentiate $P^{(L)}(D, \Lambda)$, we first rewrite its definition (7) by introducing the following notations:

$$P^{(l)}(D,\lambda) = \Psi(b^{(l-1)}(D,\lambda), D, \lambda), \tag{14}$$

$$b^{(l)}(D,\lambda) = \Phi(b^{(l-1)}(D,\lambda), D, \lambda), \tag{15}$$

where

$$\Psi(b, D, \lambda) := \prod_s \left(K^\top \frac{d_s}{Kb_s}\right)^{\lambda_s}, \tag{16}$$

$$\Phi(b, D, \lambda) := \left[\left(\frac{\Psi(b,D,\lambda)}{K^\top \frac{d_1}{Kb_1}}\right)^\top, \ldots, \left(\frac{\Psi(b,D,\lambda)}{K^\top \frac{d_S}{Kb_S}}\right)^\top\right]^\top. \tag{17}$$

Finally, we introduce the following notations for readability:

$$\xi_y^{(l)} := \left[\partial_y \xi(b^{(l)}, D, \lambda)\right]^\top, \qquad B_y^{(l)} := \left[\partial_y b^{(l)}(D,\lambda)\right]^\top,$$

where ξ can be Ψ or Φ, and y can be D or λ.

Proposition 3.1.

$$\nabla_D \mathcal{E}_L(D,\lambda) = \Psi_D^{(L-1)}\left(\nabla\mathcal{L}(P^{(L)}(D,\lambda), x)\right) + \sum_{l=0}^{L-2} \Phi_D^{(l)}\left(v^{(l+1)}\right), \tag{18}$$

$$\nabla_\lambda \mathcal{E}_L(D,\lambda) = \Psi_\lambda^{(L-1)}\left(\nabla\mathcal{L}(P^{(L)}(D,\lambda), x)\right) + \sum_{l=0}^{L-2} \Phi_\lambda^{(l)}\left(v^{(l+1)}\right), \tag{19}$$

where:

$$v^{(L-1)} := \Psi_b^{(L-1)}\left(\nabla L(P^{(L)}(D,\lambda), x)\right), \tag{20}$$

$$\forall l < L-1, v^{(l-1)} := \Phi_b^{(l-1)}\left(v^{(l)}\right). \tag{21}$$

See Appendix A proof.

4. Extensions.

4.1. Log-domain stabilization.

4.1.1. Stabilization. In its most general framework, representation learning aims at finding a useful representation of data, rather than one allowing for perfect reconstruction. In some particular cases, however, it might also be desirable to achieve a very low reconstruction error, for instance if the representation is to be used for compression of data rather than a task such as classification. In the case of our method, the quality of the reconstruction is directly linked

to the selected value of the entropy parameter γ, as it introduces a blur in the reconstructed images as illustrated in Figure 3. In the case where sharp features in the reconstructed images are desired, we need to take extremely low values of γ, which can lead to numerical problems, *e.g.* because values within the scaling vectors a and b can then tend to infinity. As suggested by Chizat et al. [17] and Schmitzer [71], we can instead perform the generalized Sinkhorn updates (7)-(8) in the log-domain. Indeed, noting $u_s^{(l)}, v_s^{(l)}$ as the dual scaling variables, that is,

$$a_s^{(l)} := \exp\left(\frac{u_s^{(l)}}{\gamma}\right), \qquad b_s^{(l)} := \exp\left(\frac{v_s^{(l)}}{\gamma}\right),$$

the quantity $-c_{ij} + u_i + v_j$ is known to be bounded and thus remains numerically stable. We can then introduce the stabilized kernel $\tilde{K}(u, v)$ defined as

$$\tilde{K}(u, v) := \exp\left(\frac{-C + u\mathbb{1}^\top + \mathbb{1}v^\top}{\gamma}\right), \tag{22}$$

and notice that we then have

$$\begin{aligned} u_s^{(l)} &= \gamma\left[\log(d_s) - \log(Kb_s^{(l-1)})\right], \\ \left[\log(Kb_s^{(l-1)})\right]_i &= \log\left[\sum_j \exp\left(\frac{-c_{ij} + v_j^{(l-1)}}{\gamma}\right)\right] \\ &= \log\left(\sum_j \tilde{K}(u_s^{(l-1)}, v_s^{(l-1)})_{.j}\right) - \frac{\left[u_s^{(l-1)}\right]_i}{\gamma}. \end{aligned}$$

With similar computations for the v_s updates, we can then reformulate the Sinkhorn updates in the stabilized domain as

$$u_s^{(l)} := \gamma\left[\log(d_s) - \log\left(\sum_j \tilde{K}(u_s^{(l-1)}, v_s^{(l-1)})_{.j}\right)\right] + u_s^{(l-1)}, \tag{23}$$

$$v_s^{(l)} := \gamma\left[\log(P^{(l)}) - \log\left(\sum_i \tilde{K}(u_s^{(l)}, v_s^{(l-1)})_{i.}\right)\right] + v_s^{(l-1)}. \tag{24}$$

This provides a forward scheme for computing Wasserstein barycenters with arbitrarily low values of γ, which could be expanded to the backward loop of our method either by applying an automatic differentiation tool to the stabilized forward barycenter algorithm or by changing the steps in the backward loop of Algorithm 1 to make them rely solely on stable quantities. However, this would imply computing a great number of stabilized kernels as in (22), which relies on nonseparable operations. Each of those kernels would also have to either be stored in memory or recomputed when performing the backward loop. In both cases, the cost in memory or number of operations, respectively, can easily be too high in large scale settings.

4.1.2. Separable log kernel. These issues can be avoided by noticing that when the application of the kernel K is separable, this operation can be performed at a much lower cost. For a d-dimensional histogram of $N = n^d$ bins, applying a separable kernel amounts to performing a sequence of d steps, where each step computes n operations per bin. It results in a $O(n^{d+1}) = O(N^{\frac{d+1}{d}})$ cost instead of $O(N^2)$. As mentioned previously, the stabilized kernel (22) is not separable, prompting us to introduce a new stable and separable kernel suitable for log-domain processing. We illustrate this process using 2-dimensional kernels without loss of generality. Let $\mathcal{X}$ be a 2-dimensional domain discretized as an $n \times n$ grid. Applying a kernel of the form $K = \exp(-\frac{C}{\gamma})$ to a 2-dimensional image $b \in \mathcal{X}$ is performed as such:

$$R(i,j) := \sum_{k=1}^{n}\sum_{l=1}^{n} \exp\left(-\frac{C((i,j),(k,l))}{\gamma}\right) b(k,l)\,,$$

where $C((i,j),(k,l))$ denotes the cost to transport mass between the points (i,j) and (k,l).

Assuming a separable cost such that $C((i,j),(k,l)) := C_y(i,k) + C_x(j,l)\,$, it amounts to performing two sets of 1-dimensional kernel applications:

$$A(k,j) = \sum_{l=1}^{n} \exp\left(\frac{C_x(j,l)}{\gamma}\right) b(k,l),$$
$$R(i,j) = \sum_{k=1}^{n} \exp\left(\frac{C_y(i,k)}{\gamma}\right) A(k,j)\,.$$

In order to stabilize the computation and avoid reaching representation limits, we transfer it to the log-domain ($v := \log(b)$). Moreover, we shift the input values by their maximum and add it at the end. The final process can be written as the operator $K_{LS} : \log(b) \to \log(K(b))$ with K a separable kernel, and is described in Algorithm 3.

Algorithm 3 `LogSeparableKer` K_{LS}: Application of a 2-dimensional separable kernel in log-domain

Inputs: Cost matrix $C \in \mathbb{R}^{N\times N}$, image in log-domain $v \in \mathbb{R}^{n\times n}$
$\forall k,j,\ x_l(k,j) := \frac{C_x(j,l)}{\gamma} + v(k,l)$
$\forall k,j,\ A'(k,j) := \log\left(\sum_l^n \exp(x_l - \max_l x_l)\right) + \max_l x_l$
$\forall i,j,\ y_k(i,j) := \frac{C_y(i,k)}{\gamma} + A'(k,j)$
$\forall i,j,\ R'(i,j) := \log\left(\sum_k^n \exp(y_k - \max_k y_k)\right) + \max_k y_k$
Outputs: Image in log-domain $K_{LS}(v) = R'$

This operator can be used directly in the forward loop, as seen in Algorithm 4. For backward loops, intermediate values can be negative and real-valued logarithms are not suited. While complex-valued logarithms solve this problem, they come at a prohibitive computational cost. Instead, we store the sign of the input values and compute logarithms of absolute values. When exponentiating, the stored sign is used to recover the correct value.

4.2. Warm start. Warm start, often used in optimization problems, consists in using the solution of a previous optimization problem, close to the current one, as the initialization point in order to speed up the convergence. Our method relies on performing an iterative optimization process (for example, L-BFGS in the following experiments) which, at each iteration, calls upon *another* iterative scheme: the forward Sinkhorn loop to compute the barycenter and its automatic differentiation to obtain gradients. As described in subsection 2.2, this second, nested iterative scheme is usually initialized with constant scaling vectors. However, in our case, since each iteration of our descent method performs a new Sinkhorn loop, the scaling vectors of the previous iteration can be used to set the values of $b^{(0)}$ instead of the usual $\mathbb{1}_{NS}$, thus "warm-starting" the barycenter computation. In the remainder of this subsection, for illustrative purposes, we will focus on our particular case where the chosen descent method is L-BFGS, though the idea of applying warm start to the generalized Sinkhorn algorithm should be directly applicable with any other optimization scheme.

As an example, in our case, instead of a single L-BFGS step after $L = 500$ Sinkhorn iterations, we perform an L-BFGS step every $L = 10$ iterations, initializing the scaling vectors as the ones reached at the end of the previous 10. This technique accumulates the Sinkhorn iterations as we accumulate L-BFGS steps. This has several consequences: a gain in precision and time, a potential increase in the instability of the scaling vectors, and changes in the energy we minimize.

First, the last scaling vectors of the previous overall iteration are closer to that of the current one than a vector of constant value. Therefore, the Sinkhorn algorithm converges more rapidly, and the final barycenters computed at each iteration gain accuracy compared to the classical version of the algorithm.

Second, as mentioned in subsection 4.1, the scaling vectors may become unstable when computing a large number of iterations of the Sinkhorn algorithm. When using a warm start strategy, Sinkhorn iterations tend to accumulate, which may consequently degrade the stability of the scaling vectors. For example, using 20 Sinkhorn iterations running through 50 L-BFGS steps, a warm start would lead to barycenters computed using scaling vectors comparable to those obtained after 1000 Sinkhorn iterations. When instabilities become an issue, we couple the warm start approach with our log-domain stabilization. The reduced speed of log-domain computations is largely compensated by the fact that our warm start allows the computation of fewer Sinkhorn iterations for an equivalent or better result.

Third, when differentiating (10), we consider the initial, warm-started (as opposed to initializing $b^{(0)}$ to $\mathbb{1}_{NS}$) values given to the scaling vectors to be constant and independent of weights and atoms. This amounts to considering a different energy to minimize at each L-BFGS step.

We demonstrate the benefits of the warm start in Figure 4. We plot the evolution of the mean peak signal-to-noise ratio (PSNR) of the reconstructions throughout the L-BFGS iterations for different settings for the two datasets used in subsection 5.4. For these examples, we used the KL loss (since it gave the best reconstructions overall), we did not have to use the log-domain stabilization, and we restarted L-BFGS every 10 iterations. At an equal number of Sinkhorn iterations (L), enabling the warm start always yields better reconstructions after a certain number of iterations. It comes at a small overhead cost in time (around 25%) because the L-BFGS line search routine requires more evaluations at the start. For the example in

Figure 4a, the computation times are 20 minutes for $L = 2$, 25 minutes for the warm restart and $L = 2$, and 15 hours for $L = 100$. In this particular case, enabling the warm start with two Sinkhorn iterations yields even better results than having 100 Sinkhorn iterations without a warm start and with a 36 gain factor in time. For the second dataset (Figure 4b), enabling the warm start does not yield results as good as when running 100 Sinkhorn iterations. However, it would require considerably more than two Sinkhorn iterations, and hence a lot more time, to achieve the same result without it. The computation times in all three cases are similar to the previous example.

4.3. Sinkhorn heavyball. As part of a generalization of the Sinkhorn algorithm for solving OT between tensor fields [59], Peyré et al. introduced relaxation variables. In the particular case of scalar OT (our framework in the present work), these relaxation variables amount to an averaging step in the Sinkhorn updates; for instance, in the case of the barycenter scaling updates (6), (8),

$$\tilde{a}_s^{(l)} = \frac{d_s}{K b_s^{(l-1)}}, \tag{25}$$

$$a_s^{(l)} = \left(a_s^{(l-1)}\right)^{\tau} \left(\tilde{a}_s^{(l)}\right)^{1-\tau},$$

$$\tilde{b}_s^{(l)} = \frac{P^{(l)}\left(D, \lambda\right)}{K^{\top} a_s^{(l)}}, \tag{26}$$

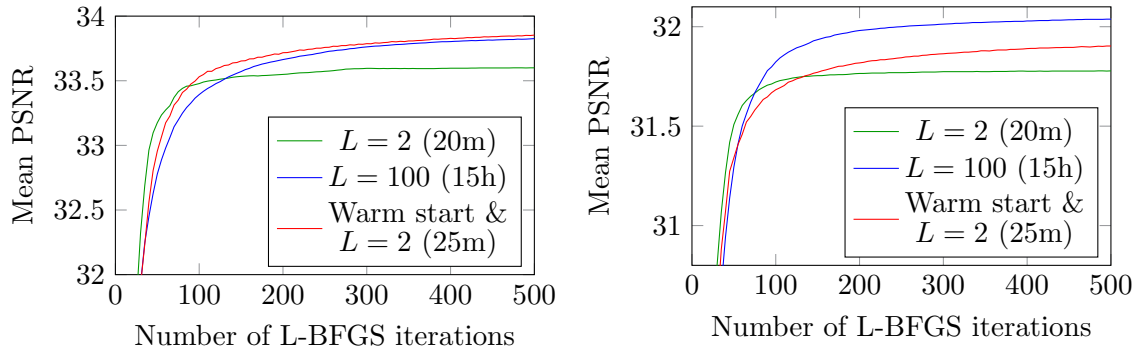
$$b_s^{(l)} = \left(b_s^{(l-1)}\right)^{\tau} \left(\tilde{b}_s^{(l)}\right)^{1-\tau}.$$

$\tau = 0$ yields the usual Sinkhorn iterations, but it has been shown that negative values of τ produce extrapolation and can lead to a considerable increase in the rate of convergence of the Sinkhorn algorithm [59, Remark 6]. This effect can be thought of in the same way as the heavy ball method [53, 87], often used in optimization problems and dating back to Polyak [61], *i.e.* as the addition of a momentum term (*e.g.*, $(a_s^{(l-1)}/\tilde{a}_s^{(l)})^{\tau}$, which amounts to $\tau(u_s^{(l-1)} - \tilde{u}_s^{(l)})$ in the log-domain) to the usual Sinkhorn updates. This acceleration scheme can be used within our method by applying an automatic differentiation tool [80] to the forward Sinkhorn loop yielding the barycenter (shown in Algorithm 5 in the Appendix) and feeding the gradients to Algorithm 2.

4.4. Unbalanced. In (1), we defined the set of admissible transport plans $\Pi(p, q)$ as the set of matrices whose marginals are equal to the two input measures, that is, with rows summing to p and columns summing to q. Equivalently, we can reformulate the definition of the approximate Wasserstein distance (3) as

$$W_{\gamma}(p, q) := \min_{T \in \mathbb{R}_+^{N \times N}} \langle T, C \rangle + \gamma H(T) + \iota_{\{p\}}(T \mathbb{1}_N) + \iota_{\{q\}}(T^{\top} \mathbb{1}_N),$$

where ι is the indicator function defined in (2). Chizat et al. introduce the notion of unbalanced transport problems [17], wherein this equality constraint between the marginals of the OT



(a) **MUG dataset: woman**

(b) **MUG dataset: man**

Figure 4: Evolution of the mean PSNR of the reconstructions per L-BFGS iteration, for three configurations, on two datasets. The KL loss was used for this experiment. We see that the warm start yields better reconstructions with the same number of Sinkhorn iterations (L) in roughly the same time.

plan and the input measures is replaced by some other similarity criterion. Using entropic regularization, they introduce matrix scaling algorithms generalizing the Sinkhorn algorithm to compute, among others, unbalanced barycenters. This generalizes the notion of approximate Wasserstein barycenters that we have focused on thus far.

In particular, using the KL divergence between the transport plan's marginals and the input measures allows for less stringent constraints on mass conservation, which can in turn yield barycenters which maintain more of the structure seen in the input measures. This amounts to using the following definition of W_γ in the barycenter formulation (5):

$$W_\gamma(p,q) := \min_{T \in \mathbb{R}_+^{N\times N}} \langle T, C\rangle + \gamma H(T) + \rho\left(\mathrm{KL}(T\mathbb{1}_N|p) + \mathrm{KL}(T^\top \mathbb{1}_N|q)\right),$$

where $\rho > 0$ is the parameter determining how far from the balanced OT case we can stray, with $\rho = \infty$ yielding the usual OT formulation. In this case, the iterative matrix scaling updates (7)–(8) read, respectively [17], as

$$P^{(l)}(D,\lambda) = \left(\sum_{s=1}^{S} \lambda_s \left(K^\top a_s^{(l)}\right)^{\frac{\gamma}{\rho+\gamma}}\right)^{\frac{\rho+\gamma}{\gamma}},$$
$$a_s^{(l)} = \left(\tilde{a}_s^{(l)}\right)^{\frac{\rho}{\rho+\gamma}}, b_s^{(l)} = \left(\tilde{b}_s^{(l)}\right)^{\frac{\rho}{\rho+\gamma}},$$

where $\tilde{a}_s^{(l)}, \tilde{b}_s^{(l)}$ are obtained from the usual Sinkhorn updates as in (25), (26).

Algorithm 6, given in the Appendix, performs the barycenter computation (forward loop) including both the unbalanced formulation and the acceleration scheme shown in subsection 4.3.

Automatic differentiation can then be performed using an appropriate library [80] to obtain the dictionary and weights gradients, which can then be plugged into Algorithm 2 to obtain a representation relying on unbalanced barycenters.

5. Applications.

5.1. Comparison with Wasserstein principal geodesics. As mentioned in subsection 1.1, an approach to generalize PCA to the set of probability measures on some space, endowed with the Wasserstein distance, has recently been proposed [74]. Given a set of input measures, an approximation of their Wasserstein principal geodesics (WPG) can be computed, namely geodesics that pass through their isobarycenter (in the Wasserstein sense) and are close to all input measures. Because of the close link between Wasserstein geodesics and the Wasserstein barycenter, it would stand to reason that the set of barycenters of $S = 2$ atoms learned using our method could be fairly close to the first WPG. In order to test this, and to compare both approaches, we reproduce the setting of the WPG paper [74] experiment on the MNIST dataset within our framework.

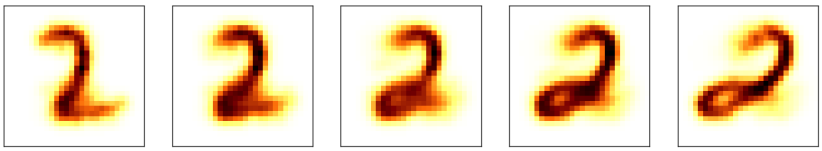


Figure 5: Span of our 2-atom dictionary for weights $(1-t, t), t \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$, when trained on images of digit 2.

We first run our method to learn two atoms on samples of 1000 images for each of the first four nonzero digits, with parameters $\gamma = 2.5, L = 30$, and compare the geodesic that runs in between the two learned atoms with the first WPG. An example of the former is shown in Figure 5. Interestingly, in this case, as with the 3's and 4's, the two appear visually extremely close (see the first columns of [74, Figure 5] for the first WPG). It appears our method *can* thus capture WPGs. We do not seem to recover the first WPG when running on the dataset made up of 1's, however. This is not unexpected, as several factors can cause the representation we learn to vary from this computation of the first WPG:

- In our case, there is no guarantee the isobarycenter of all input measures lies within the span of the learned dictionary.
- Even when it does, since we minimize a non-convex function, the algorithm might converge toward another local minimum.
- In this experiment, the WPGs are computed using several approximations [74], including some for the computation of the geodesics themselves, which we are not required to make in order to learn our representation.

Note that in the case of this particular experiment (on a subsample of MNIST 1's), we tried relaunching our method several times with different random initializations and never observed a span similar to the first WPG computed using these approximations.

Our approach further enables us to combine, in a straightforward way, each of the captured variations when learning more than two atoms. This is illustrated in Figure 6, where we run

our method with $S = 3$. Warpings similar to those captured when learning only $S = 2$ atoms (the appearance of a loop within the 2) are also captured, along with others (shrinking of the vertical size of the digit toward the right). Intermediate values of the weight given to each of the three atoms allow our representation to cover the whole simplex, thus arbitrarily combining any of these captured warpings (*e.g.*, vertically shrinked, loopless 2 in the middle of the bottom row).

Figures similar to Figure 5 and 6 for all other digits are given in the Appendix, subsection E.1.

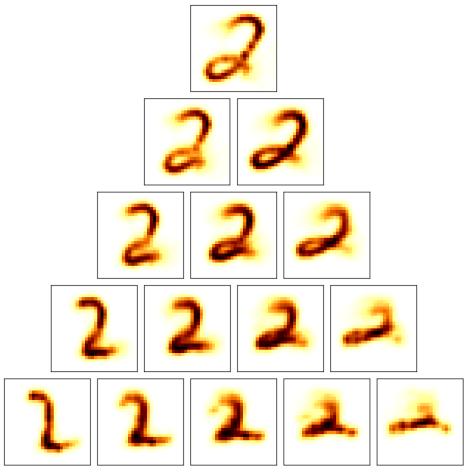
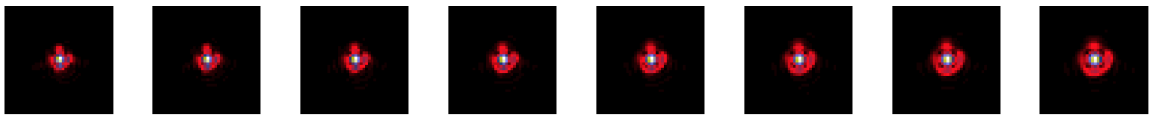
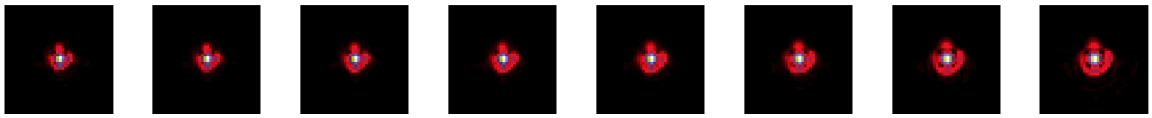


Figure 6: Span of a 3-atom dictionary learned on a set of 2's. Weights along each edge are the same as in Figure 5 for the two extreme vertices and 0 for the other, while the three center barycenters have a weight of $\frac{1}{2}$ for the atom corresponding to the closest vertex and $\frac{1}{4}$ for each of the other two.



(a) 550nm (b) 600nm (c) 650nm (d) 700nm (e) 750nm (f) 800nm (g) 850nm (h) 900nm

Figure 7: Simulated Euclid-like PSF variation at a fixed position in the field of view for varying incoming wavelengths.



(a) 550nm (b) 600nm (c) 650nm (d) 700nm (e) 750nm (f) 800nm (g) 850nm (h) 900nm

Figure 8: Polychromatic variations of PSFs by displacement interpolation.

5.2. Point spread functions. As with every optical system, observations from astrophysical telescopes suffer from a blurring related to the instrument's optics and various other effects (such as the telescope's jitter for space-based instruments). The blurring function, or point spread function (PSF), can vary spatially (across the instrument's field of view), temporally and chromatically (with the incoming light's wavelength). In order to reach its scientific goals, the European Space Agency's upcoming Euclid space mission [40] will need to measure the shape of one billion galaxies extremely accurately, and therefore correcting the PSF effects is of paramount importance. The use of OT for PSF modeling has been investigated by Irace and Batatia [38] and Ngolè and Starck [54], both with the aim of capturing the spatial variation of the PSF. For any given position in the field of view, the transformations undergone by the PSF depending on the incoming light's wavelength are also known to contain strong geometrical information, as illustrated in Figure 7. It is therefore tempting to express these variations as the intermediary steps in the optimal transportation between the PSFs at the two extreme wavelengths. This succession of intermediary steps, the *displacement interpolation* (also known as McCann's interpolation [48]) between two measures, can be computed (in the case of the 2-Wasserstein distance) as their Wasserstein barycenters with weights $\lambda = (1-t, t), t \in [0, 1]$ [1].

We thus ran our method on a dataset of simulated, Euclid-like PSFs [55, §4.1] at various wavelengths and learned only two atoms. The weights were initialized as a projection of the wavelengths into $[0, 1]$ but allowed to vary. The atoms were initialized without using any prior information as two uniform images with all pixels set at $1/N$, N being the number of pixels (in this case 40^2). The fitting loss was quadratic, the entropic parameter γ set to a value of 0.5 to allow for sharp reconstructions, and the number of Sinkhorn iterations set at 120, with a heavyball parameter $\tau = -0.1$.

The learned atoms, as well as the actual PSFs at both ends of the spectrum, are shown in Figure 9. Our method does indeed learn atoms that are extremely close visually to the

two extremal PSFs. The reconstructed PSFs at the same wavelength as those of Figure 7 are shown in Figure 8 (the corresponding final barycentric weights are shown in Figure 11b). This shows that OT, and in particular displacement interpolation, does indeed capture the geometry of the polychromatic transformations undergone by the PSF. On the other hand, when one learns only two components using a PCA, they have no direct interpretation (see Figure 10), and the weights given to the 2nd principal component appear to have no direct link to the PSF's wavelength, as shown in Figure 11a.

Note that while adding constraints can also make linear generative methods yield two atoms that are visually close to the extreme PSFs, for instance by using NMF instead of PCA (see Figure E.5), our method yields lower reconstruction error, with an average normalized mean square error of 1.71×10^{-3} across the whole dataset, as opposed to 2.62×10^{-3} for NMF. As expected, this difference in reconstruction error is particularly noticeable for datapoints corresponding to wavelengths in the middle of the spectrum, as the NMF reconstruction then simply corresponds to a weighted sum of the two atoms, while our method captures more complex warping between them. This shows that the OT representation allows us to better capture the nonlinear geometrical variations due to the optical characteristics of the telescope.

5.3. Cardiac sequences. We tested our dictionary learning algorithm on a reconstructed MRI sequence of a beating heart. The goal was to learn a dictionary of four atoms, representing the key frames of the sequence.

An advantageous side effect of the weights learned by our method lying in the simplex is that it provides a natural way to visualize them: by associating each atom d_i with a fiducial position $(x_i, y_i) \in \mathbb{R}^2$, each set of weights can be represented as one point placed at the position of the Euclidean barycenter of these positions, with individual weights given to the corresponding atom. Up to rotations and inverse ordering, there are only as many such representations as there are possible orderings of the atoms. In the present case of $S = 4$, we can further use the fact that any of the four weights λ_i is perfectly known through the other three as $1 - \sum_{j \neq i} \lambda_j$. By giving atoms fiducial positions in $\mathbb{R}^3$ and ignoring one of them or, equivalently, assigning it the $(0, 0, 0)$ position, we thus obtain a unique representation of the weights as seen in Figure 12. The "barycentric path" (polyline of the barycentric points) is a cycle, which means the algorithm is successful at finding those key frames that, when interpolated, can represent the whole dataset. This is confirmed by the similarity between the reconstructions and the input measures.

For this application, we used 13 frames of 272×240, a regularization $\gamma = 2$, and a scale between weights and atoms of $\zeta = N/(100 * M)$, $N = 272 \times 240$, $M = 13$ frames. Initialization was random for the weights, and constant for the atoms. We used a quadratic loss because it provided the best results in terms of reconstruction and representation. We found 25 iterations for the Sinkhorn algorithm to be a good trade-off between computation time and precision.

5.4. Wasserstein faces. It has been shown that images of faces, when properly aligned, span a low-dimensional space that can be obtained via PCA. These principal components, called Eigenfaces, have been used for face recognition [81]. We show that, with the right setting, our dictionary learning algorithm can produce atoms that can be interpreted more easily than their linear counterparts, and can be used to edit a human face's appearance.

We illustrate this application on the MUG facial expression dataset [3]. From the raw

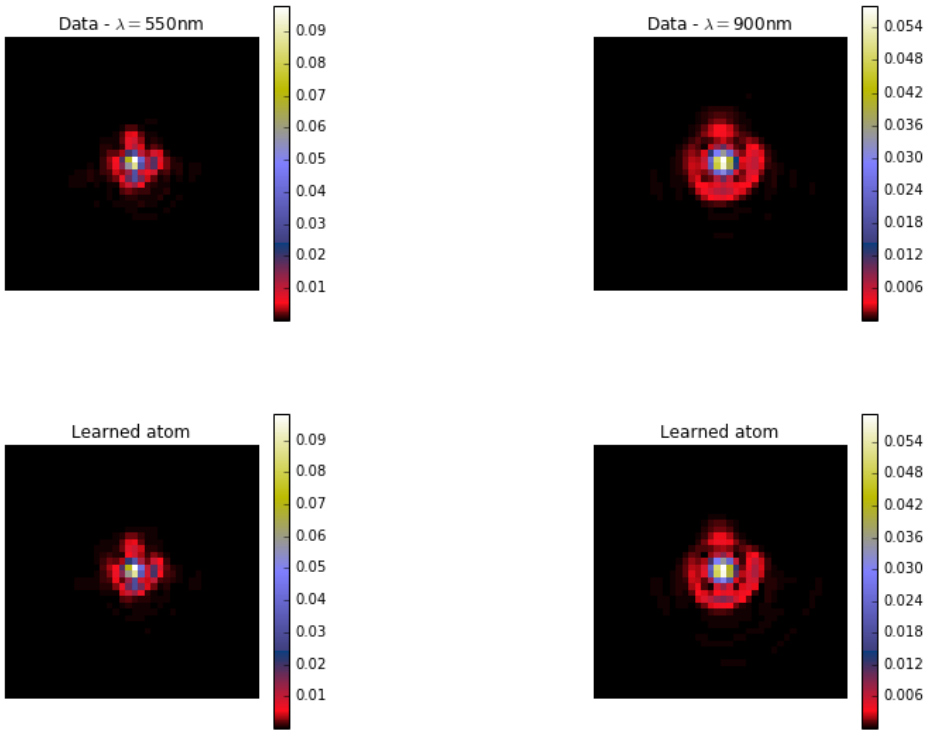


Figure 9: Extreme wavelength PSFs in the dataset and the atoms making up the learned dictionary.

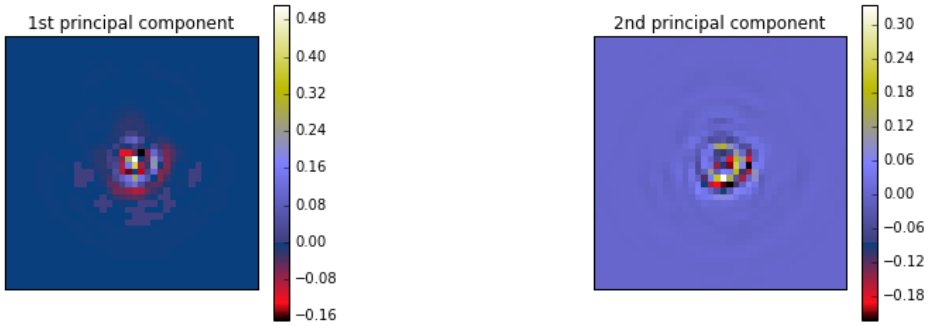
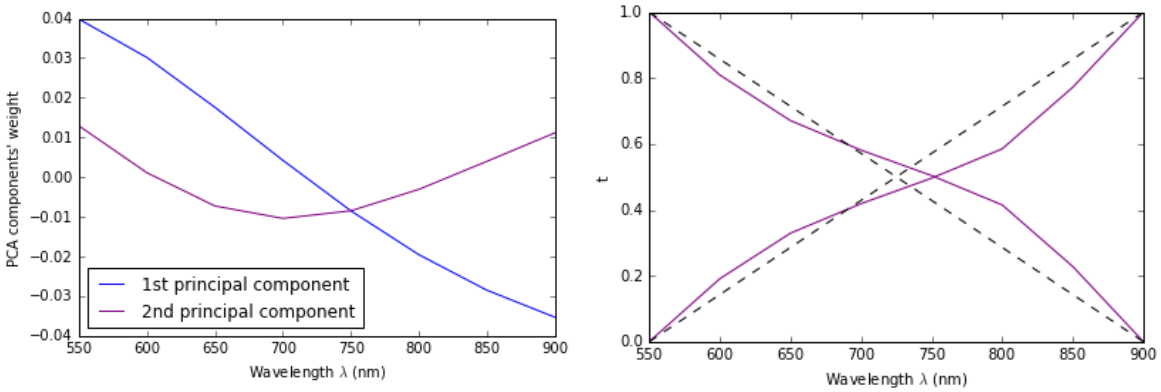


Figure 10: PCA-learned components.



(a) Weights for the first two principal components learned by PCA.

(b) Barycentric weights learned by our method. The dashed lines are the initialization.

Figure 11: Evolution of representation coefficients by wavelength.

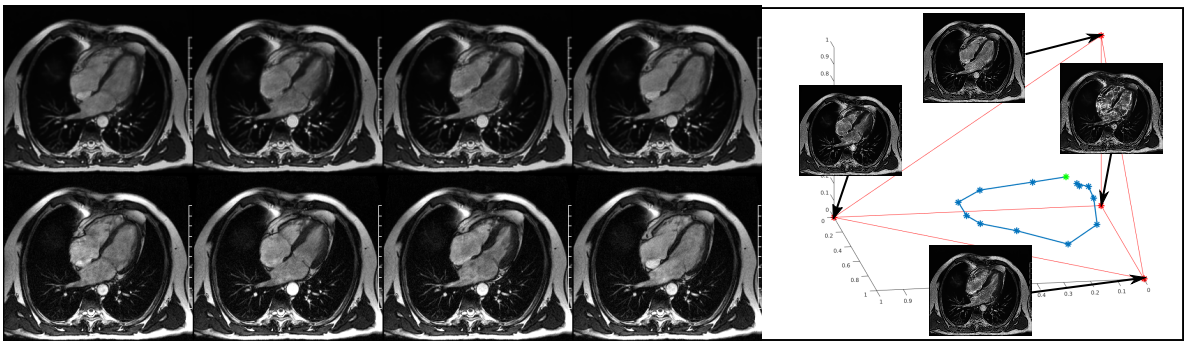


Figure 12: Left: Comparison between fpur frames (out of 13) of the measures (lower row) and the same reconstructed frames (upper row). Right: plot of the reconstructed frames (blue points) by their barycentric coordinates in the 4-atom basis, with each atom (red points) at the vertices of the tetrahedra. The green point is the first frame.

images of the MUG database, we isolated faces and converted the images to grayscale. The resulting images are in Figure 13(a). We can optionally invert the colors and apply a power factor α similarly to a gamma-correction. We used a total of 20 (224×224) images of a single person performing five facial expressions and learned dictionaries of five atoms using PCA, NMF, a K-SVD implementation [66], and our proposed method. For the last, we set the number of Sinkhorn iterations to 100 and the maximum number of L-BFGS iterations to 450. The weights were randomly initialized, and the atoms were initialized as constant.

We performed a cross validation using two datasets, four loss functions, four values for α $(1, 2.2, 3, 5)$, and colors either inverted or not. We found that none of the α values we tested gave significantly better results (in terms of reconstruction errors). Interestingly, however,

inverting colors improved the result for our method in most cases. We can conclude that when dealing with faces, it is better to transport the thin and dark zones (eyebrows, mouth, creases) than the large and bright ones (cheeks, forehead, chin).

As illustrated by Figure 13 (and E.7 in the Appendix), our method reaches similarly successful reconstructions given the low number of atoms, with a slightly higher mean PSNR of 33.8 compared to PSNRs of 33.6, 33.5 and 33.6 for PCA, NMF and K-SVD respectively.

We show in Figure 14 (and E.6 in Appendix) the atoms obtained when using different loss functions. This shows how sensible the learned atoms are to the chosen fitting loss, which highlights the necessity for its careful selection if atoms' interpretability is important for the application at hand.

Finally, we showcase an appealing feature of our method: the atoms that it computes allow for facial editing. We demonstrate this application in Figure 15. Starting from the isobarycenter of the atoms, by interpolating weights towards a particular atom, we add some of the corresponding emotion to the face.

5.5. Literature learning. We use our algorithm to represent literary work. To this end, we use a bag-of-words representation [68], where each book is represented by a histogram of its words. In this particular application, the cost matrix C (distance between each word) is computed exhaustively and stored. We use a semantic distance between words. These distances were computed from the Euclidian embedding provided by the GloVe database (Global Vectors for Word Representation) [58].

Our learning algorithm is unsupervised and considers similarity between books based on their lexical fields. Consequently we expect it to sort books by either author, writing style, or genre.

To demonstrate our algorithm's performance, we created a database of 20 books by five different authors. In order to keep the problem size reasonable we only considered words that are between seven and eight letters long. In our case, it is better to deal with long words because they have a higher chance of holding discriminative information than shorter ones.

The results can be seen in Figure 16. Our algorithm is able to group the novels by author, recognizing the proximity of lexical fields across the different books. Atom 0 seems to be representing Charlotte Brontë's style, atoms 1 and 4 that of Mark Twain, atom 2 that of Arthur Conan Doyle, and atom 3 that of Jane Austen. Charles Dickens appears to share an extended amount of vocabulary with the other authors without it differing enough to be represented by its own atom, like others are.

5.6. Multimodal distributions. It is a well-known limitation of the regular OT-based Wasserstein barycenters that when there are several distinct areas containing mass, the supports of which are disjoint on the grid, the barycenter operator will still produce barycenters with mass in between them. To illustrate the advantages of using the unbalanced version our method introduced in subsection 4.4 and the use cases where it might be preferable to do so, we place ourselves in such a setting.

We generate a dataset as follows: A 1-dimensional grid is separated into three equal parts, and while the center part is left empty, we place two discretized and truncated 1-dimensional Gaussians with the same standard deviation, their mean randomly drawn from every other appropriate position on the grid. We draw 40 such datapoints, yielding several distributions

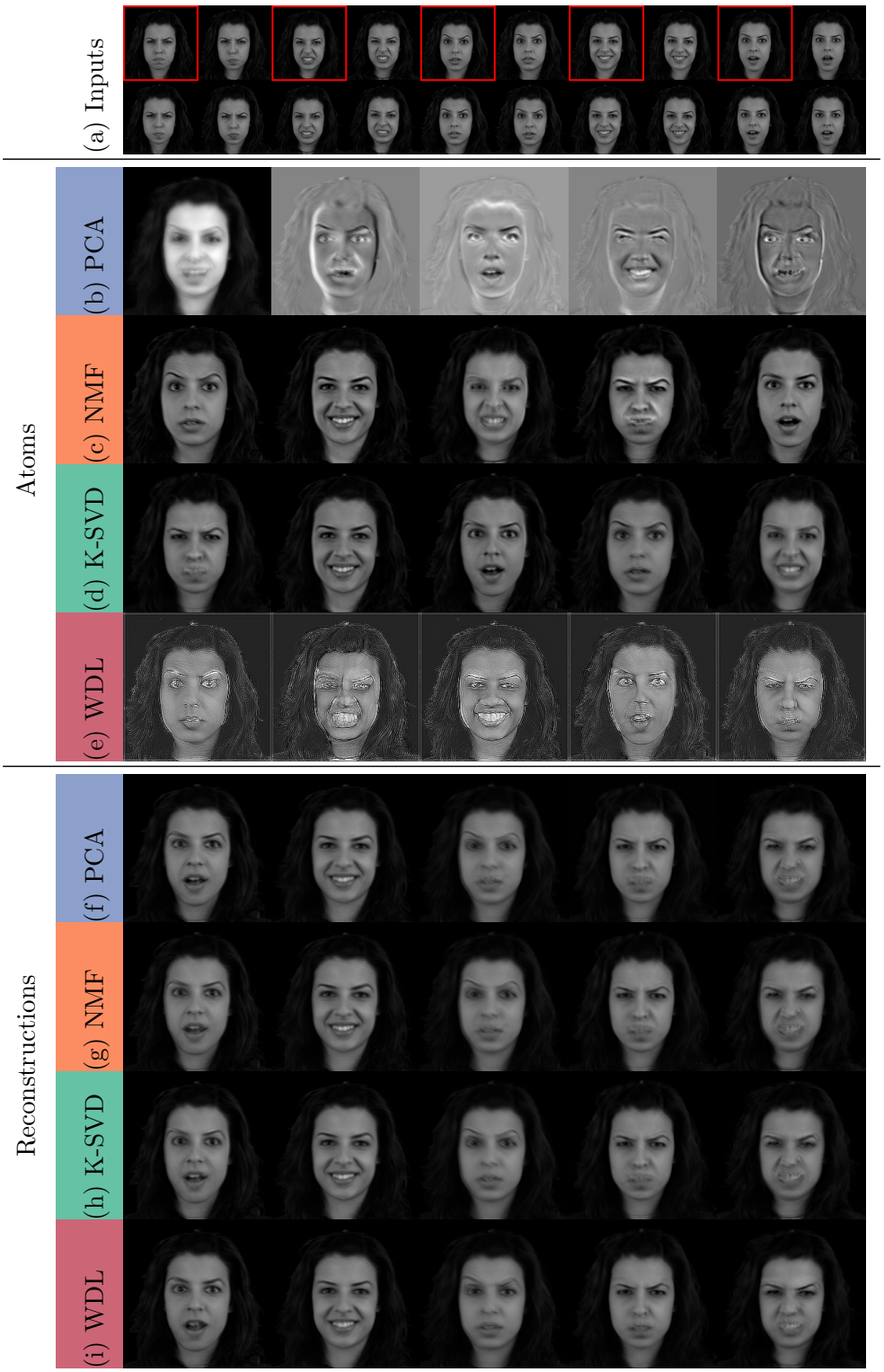


Figure 13: We compare our method with Eigenfaces [81], NMF and K-SVD [66] as a tool to represent faces on a low-dimensional space. Given a dataset of 20 images of the same person from the MUG dataset [3] performing five facial expressions four times (row (a) illustrates each expression), we project the dataset on the first five Eigenfaces (row (b)). The reconstructed faces corresponding to the highlighted input images are shown in row (f). Rows (c) and (d), respectively, show atoms obtained using NMF and K-SVD and rows (g) and (h) their respective reconstructions. Using our method, we obtain five atoms shown in row (e) that produce the reconstructions in row (i).

Figure 14: We compare the atoms (columns 1 to 5) obtained using different loss functions, ordered by the fidelity of the reconstructions to the input measures (using the mean PSNR), from best to worst: the KL divergence (a) $\overline{PSNR} = 32.03$, the quadratic loss (b) $\overline{PSNR} = 31.93$, the total variation loss (c) $\overline{PSNR} = 31.41$, and the Wasserstein loss (d) $\overline{PSNR} = 30.33$. In the last column, we show the reconstruction of the same input image for each loss. We notice that from (a) to (d), the atoms' visual appearance seems to increase even though the reconstruction quality decreases.

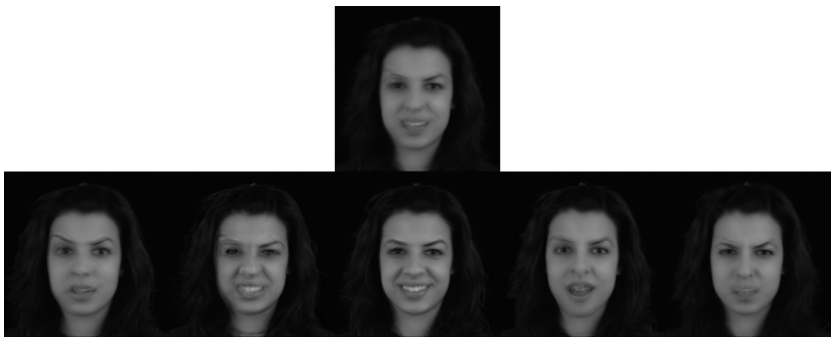


Figure 15: Face editing : Using the atoms shown in row (a) of Figure E.6, we interpolate between the atoms' isobarycenter (top image) and each one of the atoms (giving it a relative contribution of 70%). This allows us to emphasize each emotion (bottom images) when starting from a neutral face.

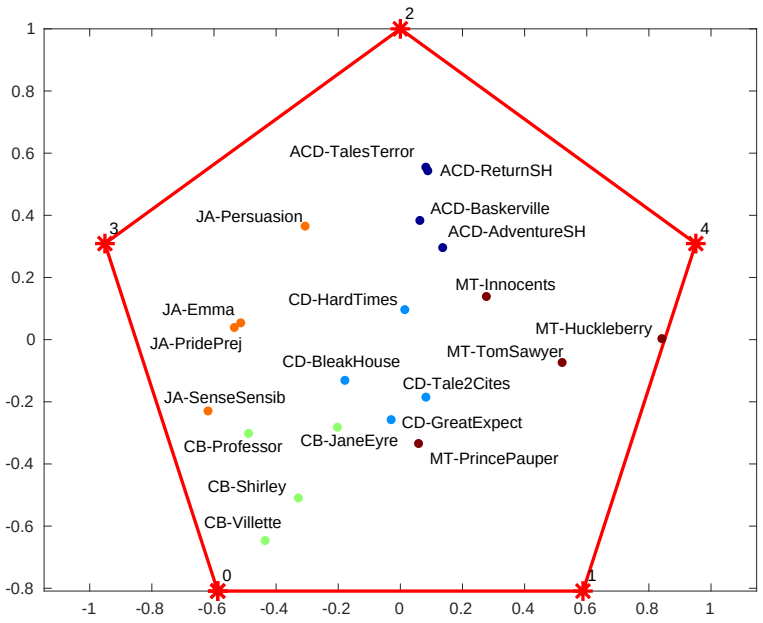


Figure 16: Using our algorithm, we look at word histograms of novels and learn five atoms in a sample of 20 books by five authors. Each book is plotted according to its barycentric coordinates with regard to the learned atoms, as explained in subsection 5.3.

with either one (if the same mean is drawn twice) or two modes in one of the two extreme parts of the grid or one mode in each.

We then run our method in both the balanced and the unbalanced settings. In both cases, γ is set to 7, 100 Sinkhorn iterations are performed, the loss is quadratic, and the learned dictionary is made up of three atoms. In the unbalanced case, the KL-regularization parameter is set as $\rho = 20$.

Figure 17 shows examples of the input data and its reconstructions in both settings. In the unbalanced case, our method always yields the right number of modes in the right parts of the grid. Running our method with balanced Wasserstein barycenters, however, leads to reconstructions featuring mass in parts of the grid where there was none in the original datapoint (the two left-most examples). Parts of the grid where the datapoint featured a mode can also be reconstructed as empty (the third example). Lastly, we observe mass in areas of the grid that were empty for *all datapoints* (the fourth example).

6. Conclusion. This paper introduces a nonlinear dictionary learning approach that uses OT geometry by fitting data to Wasserstein barycenters of a list of learned atoms. We offer schemes to compute this representation based on the addition of an entropic penalty to the definition of OT distances, as well as several variants and extensions of our method. We illustrate the representation our approach yields on several different applications.

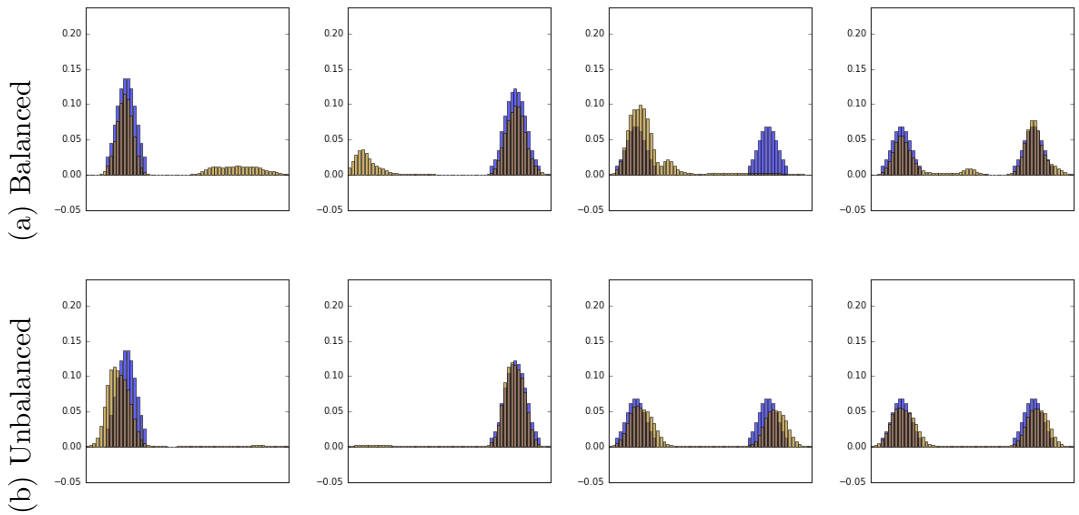


Figure 17: Four different original datapoints (in blue) and their reconstructions (in yellow) from our method in both the balanced (top row) and unbalanced (bottom row) settings. In the balanced case, we see the appearance of spurious modes where there was no mass in the original data or a lack of mass where there was a mode originally (the third example). Conversely, in the unbalanced case, our approach always places mass at the right positions on the grid.

Some very recent works present a faster Sinkhorn routine, such as the Greenkhorn algorithm [4] or a multiscale approach [71]. These could be integrated into our method along with automatic differentiation in order to speed up the algorithm.

Appendix A. Proof of Proposition 3.1. By differentiating (14) with regard to the dictionary or one of the barycentric weights, we can rewrite the Jacobians in (11), (12), respectively, while separating the differentiations with regard to the dictionary D, the weights λ_i and the scaling vector b, by total differentiation and the chain rule:

$$\left[\partial_D P^{(l)}(D,\lambda)\right]^\top = \Psi_D^{(l-1)} + B_D^{(l-1)}\Psi_b^{(l-1)}, \tag{27}$$

$$\left[\partial_\lambda P^{(l)}(D,\lambda)\right]^\top = \Psi_\lambda^{(l-1)} + B_\lambda^{(l-1)}\Psi_b^{(l-1)}. \tag{28}$$

And, differentiating (15),

$$B_D^{(l)} = \Phi_D^{(l-1)} + B_D^{(l-1)}\Phi_b^{(l-1)}, \tag{29}$$

$$B_\lambda^{(l)} = \Phi_\lambda^{(l-1)} + B_\lambda^{(l-1)}\Phi_b^{(l-1)}. \tag{30}$$

We then have, by definitions (20)-(21) and by plugging (27) and (29) into (11),

$$\begin{aligned}
\nabla_D \mathcal{E}_L(D,\lambda) &= \Psi_D^{(L-1)}\left(\nabla\mathcal{L}(P^{(L)}(D,\lambda),x)\right) + B_D^{(L-1)}v^{(L-1)} \\
&= \Psi_D^{(L-1)}\left(\nabla\mathcal{L}(P^{(L)}(D,\lambda),x)\right) + \Phi_D^{(L-2)}\left(v^{(L-1)}\right) + B_D^{(L-2)}\left(v^{(L-2)}\right) \\
&= \dots \\
\nabla_D \mathcal{E}_L(D,\lambda) &= \Psi_D^{(L-1)}\left(\nabla\mathcal{L}(P^{(L)}(D,\lambda),x)\right) + \sum_{l=0}^{L-2}\Phi_D^{(l)}\left(v^{(l+1)}\right),
\end{aligned} \tag{31}$$

where the sum starts at 0 because $B_D^{(0)} = 0$ since we initialized $b^{(0)}$ as a constant vector. This proves (18). Similarly, differentiating with regard to λ yields

$$\nabla_\lambda \mathcal{E}_L(D,\lambda) = \Psi_\lambda^{(L-1)}\left(\nabla\mathcal{L}(P^{(L)}(D,\lambda),x)\right) + \sum_{l=0}^{L-2}\Phi_\lambda^{(l)}\left(v^{(l+1)}\right).$$

This proves (19). The detailed derivation of the differentials of φ, Φ, and Ψ with regard to all three variables is given in the Appendix, Appendix C.

Appendix B. Stabilized backward loop.

Algorithm 4 `logSinkhornGrads`: Computation of dictionary and barycentric weights gradients in log-domain. Log-domain variables are marked with a tilde.

Inputs: Data $x \in \Sigma_N$, atoms $d_1, \dots, d_S \in \Sigma_N$, current weights $\lambda \in \Sigma_S$

comment: Sinkhorn loop

$\forall s, v_s^{(0)} := \mathbf{0}_N$

for $l = 1$ to L step 1 do

$\quad \forall s, \tilde{\varphi}_s^{(l)} := K_{LS}\left(\log(d_s) - K_{LS}(v_s^{(l-1)})\right)$

$\quad \tilde{p} := \sum_s \lambda_s \tilde{\varphi}_s^{(l)}$

$\quad \forall s, v_s^{(l)} := \tilde{p} - \tilde{\varphi}_s^{(l)}$

od

$p = \exp(\tilde{p})$

comment: Backward loop - weights

$w := \mathbf{0}_S$

$r := \mathbf{0}_{S\times N}$

$g := \nabla\mathcal{L}(p, x) \odot p$

for $l = L$ to 1 step -1 do

$\quad \forall s, w_s := w_s + \langle \tilde{\varphi}_s^{(l)}, g \rangle$

$\quad \forall s, \tilde{t}_s := K_{LS}\left(\log(\lambda_s g - r_s) - \tilde{\varphi}_s^{(l)}\right) + \log(d_s) - 2 * K_{LS}(v_s^{(l-1)})$

$\quad \forall s, r_s := \exp\left(K_{LS}(\tilde{t}_s) + v_s^{(l-1)}\right)$

$\quad g := -\sum_s r_s$

od

comment: Backward loop - dictionary

$y := \mathbf{0}_{S\times N}$

$z := \mathbf{0}_{S\times N}$

$n := \nabla\mathcal{L}(p, x)$

for $l = L$ to 1 step -1 do

$\quad \forall s, \tilde{c}_s := K_{LS}\left(\log(\lambda_s n + z_s) + v_s^{(l)}\right)$

$\quad \forall s, y_s := y_s + \exp\left(\tilde{c}_s - K_{LS}(v_s^{(l-1)})\right)$

$\quad \forall s, z_s := \exp\left(-\tilde{\varphi}_s^{(l-1)} + K_{LS}\left(\log(d_s) + \tilde{c}_s - 2 * K_{LS}(v_s^{(l-1)})\right)\right)$

$\quad n := -\sum_s z_s$

od

Outputs: $P^{(L)}(D, \lambda) := p, \nabla_D \mathcal{E}^{(L)} := y, \nabla_\lambda \mathcal{E}^{(L)} := w$

Appendix C. Detailed derivations.

Let us first introduce the following notation:

$$\varphi : \begin{array}{c} \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^N \\ b_s, d \mapsto K^\top \dfrac{d}{Kb_s} \end{array}.$$

C.1. Computation of $\partial_b \varphi$. By definition,

$$\frac{\partial \varphi}{\partial b_s}(b_s, d) = -K^\top \Delta\left(\frac{d}{(Kb_s)^2}\right) K. \tag{32}$$

In what follows, we will denote $\varphi_{NS}(b, D) = \left[\varphi(b_1, d_1)^\top, \dots, \varphi(b_S, d_S)^\top\right]^\top \in \mathbb{R}^{NS}$.

$$\partial_b \varphi_{NS}(b, D) = \begin{pmatrix} \frac{\partial \varphi(b_1,d_1)}{\partial b_1} & \mathbf{0}_{N\times N} & \dots & \mathbf{0}_{N\times N} \\ \mathbf{0}_{N\times N} & \frac{\partial \varphi(b_2,d_2)}{\partial b_2} & \dots & \mathbf{0}_{N\times N} \\ \vdots & & \ddots & \vdots \\ \mathbf{0}_{N\times N} & \dots & \mathbf{0}_{N\times N} & \frac{\partial \varphi(b_S,d_S)}{\partial b_S} \end{pmatrix}.$$

C.2. Computation of Ψ_b. Taking the logarithm of (16) yields

$$\log(\Psi(b, D, \lambda)) = \sum_s \lambda_s \log(\varphi(b_s, d_s)),$$

the differentiation of which gives us

$$\begin{aligned} \Delta\left(\frac{\mathbb{1}_N}{\Psi(b, D, \lambda)}\right) \partial_b \Psi(b, D, \lambda) &= \begin{pmatrix} \lambda_1 I_N & \dots & \lambda_S I_N \end{pmatrix} \Delta\left(\frac{\mathbb{1}_{NS}}{\varphi_{NS}(b, D)}\right) \partial_b \varphi_{NS}(b, D) \\ \implies \Psi_b &= [\partial_b \varphi_{NS}(b, D)]^\top \Delta\left(\frac{\mathbb{1}_{NS}}{\varphi_{NS}(b, D)}\right) J_\lambda \Delta(\Psi(b, D, \lambda)), \end{aligned} \tag{33}$$

where $J_\lambda = \begin{pmatrix} \lambda_1 I_N \\ \vdots \\ \lambda_S I_N \end{pmatrix} \in \mathbb{R}^{NS\times N}$.

C.3. Computation of Ψ_D. Let $i \in \{1, \dots, S\}$.

$$\Psi(b, D, \lambda) = \prod_{s\neq i} \Delta(\varphi_c(b_s, d_s))^{\lambda_s} . \left(K^\top \frac{d_i}{Kb_i}\right)^{\lambda_i},$$

and

$$\begin{aligned} \frac{\partial \left(K^\top \frac{d_i}{Kb_i}\right)^{\lambda_i}}{\partial d_i} &= \lambda_i \Delta\left(K^\top \frac{d_i}{Kb_i}\right)^{\lambda_i - 1} K^\top \Delta\left(\frac{\mathbb{1}_N}{Kb_i}\right) \\ \implies \frac{\partial \Psi}{\partial d_i}(b, D, \lambda) &= \lambda_i \frac{\Delta(\Psi(b, D, \lambda))}{\Delta\left(K^\top \frac{d_i}{Kb_i}\right)} K^\top \left(\frac{\mathbb{1}_N}{Kb_s}\right). \end{aligned} \tag{34}$$

C.4. Computation of Φ_b.

$$\partial_b \Phi(b, D, \lambda) = \begin{pmatrix} \Delta\left(\frac{\mathbb{1}_N}{\varphi(b_1,d_1)}\right) \\ \vdots \\ \Delta\left(\frac{\mathbb{1}_N}{\varphi(b_S,d_S)}\right) \end{pmatrix} \partial_b \Psi(b, d)$$

$$- \begin{pmatrix} \Delta\left(\frac{\Psi(b,D,\lambda)}{\varphi(b_1,d_1)^2}\right)\frac{\partial\varphi(b_1,d_1)}{\partial b_1} & \mathbf{0}_{N\times N} & \dots & \mathbf{0}_{N\times N} \\ \mathbf{0}_{N\times N} & \Delta\left(\frac{\Psi(b,D,\lambda)}{\varphi(b_2,d_2)^2}\right)\frac{\partial\varphi(b_2,d_2)}{\partial b_2} & \dots & \mathbf{0}_{N\times N} \\ \vdots & & \ddots & \vdots \\ \mathbf{0}_{N\times N} & \dots & \mathbf{0}_{N\times N} & \Delta\left(\frac{\Psi(b,D,\lambda)}{\varphi(b_S,d_S)^2}\right)\frac{\partial\varphi(b_S,d_S)}{\partial b_S} \end{pmatrix}$$

$$= \Delta\left(\frac{\mathbb{1}_{NS}}{\varphi_{NS}(b,D)}\right) I_{N,S}^\top(\partial_b \Psi(b, D, \lambda)) - \Delta\left(\frac{\mathbb{1}_{NS}}{\varphi_{NS}(b,D)}\right)\Delta(\Phi(b, D, \lambda))\partial_b\varphi_{NS}(b, D)$$

$$= \Delta\left(\frac{\mathbb{1}_{NS}}{\varphi_{NS}(b,D)}\right)\left[I_{N,S}^\top(\partial_b \Psi(b, D, \lambda)) - \Delta(\Phi(b, D, \lambda))\partial_b\varphi_{NS}(b, D)\right]$$

$$\implies \Phi_b = \left[\Psi_b I_{N,S} - [\partial_b\varphi_{NS}(b, D)]^\top\Delta(\Phi(b, D, \lambda))\right]\Delta\left(\frac{\mathbb{1}_{NS}}{\varphi_{NS}(b,D)}\right)$$

$$\overset{(33)}{=} [[\partial_b\varphi_{NS}(b, D)]^\top\Delta\left(\frac{\mathbb{1}_{NS}}{\varphi(b,D)}\right) J_\lambda\Delta(\Psi(b, D, \lambda))I_{N,S}$$

$$- [\partial_b\varphi_{NS}(b, D)]^\top\Delta(\Phi(b, D, \lambda))]\Delta\left(\frac{\mathbb{1}_{NS}}{\varphi_{NS}(b,D)}\right)$$

(35)

$$= [\partial_b\varphi_{NS}(b, D)]^\top\left[\Delta\left(\frac{\mathbb{1}_{NS}}{\varphi(b,D)}\right) J_\lambda\Delta(\Psi(b, D, \lambda))I_{N,S} - \Delta(\Phi(b, D, \lambda))\right]\Delta\left(\frac{\mathbb{1}_N}{\varphi_{NS}(b,D)}\right),$$

where $I_{N,S} = [I_N, \dots, I_N] \in \mathbb{R}^{N\times NS}$. Moreover, we have

$$\Delta\left(\frac{\mathbb{1}_{NS}}{\varphi(b,D)}\right) J_\lambda\Delta(\Psi(b, D, \lambda)) = \begin{pmatrix} \Delta(1/\varphi(b_1, d_1)) & & \\ & \ddots & \\ & & \Delta(1/\varphi(b_S, d_S)) \end{pmatrix} \begin{pmatrix} \lambda_1\Delta(\Psi(b, D, \lambda)) \\ \vdots \\ \lambda_S\Delta(\Psi(b, D, \lambda)) \end{pmatrix}$$

$$= \begin{pmatrix} \lambda_1\Delta\left(\frac{\Psi(b,D,\lambda)}{\varphi(b_1,d_1)}\right) & & \\ & \ddots & \\ & & \lambda_S\Delta\left(\frac{\Psi(b,D,\lambda)}{\varphi(b_S,d_S)}\right) \end{pmatrix}$$

$$= \Delta(\Phi(b, D, \lambda)) \begin{pmatrix} \lambda_1 I_N \\ \vdots \\ \lambda_S I_N \end{pmatrix}$$

$$\Delta\left(\frac{\mathbb{1}_{NS}}{\varphi(b,D)}\right) J_\lambda\Delta(\Psi(b, D, \lambda)) = \Delta(\Phi(b, D, \lambda))J_\lambda.$$

Hence, in (35),

$$\Phi_b = [\partial_b \varphi_{NS}(b, D)]^\top \Delta(\Phi(b, D, \lambda))[J_\lambda I_{N,S} - I_{NS}]\Delta\left(\frac{\mathbb{1}_N}{\varphi_{NS}(b, D)}\right).$$

C.5. Computation of Φ_D. Let $i \in \{1, \dots\}$. $\forall s \neq i$, the only dependency in d_i of $\Phi^s(b, D, \lambda)$ resides in Ψ (see (17)), hence

$$\begin{aligned}
\forall s \neq i, \frac{\partial \Phi^s}{\partial d_i} &= \Delta\left(\frac{\mathbb{1}_N}{\varphi(b_s, d_s)}\right)\partial_{d_i}\Psi \\
&\overset{(34)}{=} \lambda_i \frac{\Delta(\Psi(B, D, \lambda))}{\Delta(\varphi(b_s, d_s))\Delta(\varphi(b_i, d_i))} K^\top \Delta\left(\frac{\mathbb{1}_N}{K b_i}\right) \\
&\overset{(17)}{=} \lambda_i \frac{\Delta(\Phi^i(B, D, \lambda))}{\Delta(\varphi(b_s, d_s))} K^\top \Delta\left(\frac{\mathbb{1}_N}{K b_i}\right).
\end{aligned}$$

As for $s = i$, we have

$$\begin{aligned}
\Phi^i(b, D, \lambda) &= \frac{\Psi(b, D, \lambda)}{K^\top \frac{d_i}{K b_i}} \\
\implies \frac{\partial \Phi^i}{\partial d_i}(b, D, \lambda) &= \Delta\left(\frac{\mathbb{1}_N}{\varphi(b_1, d_1)}\right)\partial_D \Psi(b, D, \lambda) - \frac{\Delta(\Psi(b, D, \lambda))}{\Delta(\varphi_i(b_i, d_i)^2)}\partial_{d_i}\varphi(b_i, d_i) \\
&= \Delta\left(\frac{\mathbb{1}_N}{\varphi(b_1, d_1)}\right)\partial_D \Psi(b, D, \lambda) - \frac{\Delta(\Phi^i(b, D, \lambda))}{\Delta(\varphi(b_i, d_i))} K^\top \left(\frac{\mathbb{1}_N}{K b_i}\right) \\
&= (\lambda_i - 1)\frac{\Delta(\Phi^i(b, D, \lambda))}{\Delta(\varphi(b_i, d_i))} K^\top \Delta\left(\frac{\mathbb{1}_N}{K b_i}\right).
\end{aligned}$$

Appendix D. Generalized barycenters.

Algorithm 5 `HeavyballSinkhorn`: Computation of approximate Wasserstein barycenters with acceleration

Inputs: Data $x \in \Sigma_N$, atoms $d_1, \dots, d_S \in \Sigma_N$, weights $\lambda \in \Sigma_S$, extrapolation parameter $\tau \leq 0$

$\forall s, b_s^{(0)} := \mathbf{1}_N$

for $l = 1$ to L step 1 do

$\forall s, \tilde{a}_s^{(l)} := \frac{d_s}{K b_s^{(l-1)}}$

$\forall s, a_s^{(l)} := \left(a_s^{(l-1)}\right)^{\tau} \left(\tilde{a}_s^{(l)}\right)^{1-\tau}$

$p := \prod_s \left(K^\top a_s^{(l)}\right)^{\lambda_s}$

$\forall s, \tilde{b}_s^{(l)} := \frac{p}{K^\top a_s^{(l)}}$

$\forall s, b_s^{(l)} := \left(b_s^{(l-1)}\right)^{\tau} \left(\tilde{b}_s^{(l)}\right)^{1-\tau}$

od

Outputs: $P^{(L)}(D, \lambda) := p$

Algorithm 6 `GeneralizedSinkhorn`: Computation of unbalanced barycenters with acceleration

Inputs: Data $x \in \Sigma_N$, atoms $d_1, \dots, d_S \in \Sigma_N$, weights $\lambda \in \Sigma_S$, extrapolation parameter $\tau \leq 0$, KL parameter $\rho > 0$

$\forall s, b_s^{(0)} := \mathbf{1}_N$

for $l = 1$ to L step 1 do

$\forall s, \tilde{a}_s^{(l)} := \left(\frac{d_s}{K b_s^{(l-1)}}\right)^{\frac{\rho}{\rho+\gamma}}$

$\forall s, a_s^{(l)} := \left(a_s^{(l-1)}\right)^{\tau} \left(\tilde{a}_s^{(l)}\right)^{1-\tau}$

$p := \left(\sum_{s=1}^S \lambda_s \left(K^\top a_s^{(l)}\right)^{\frac{\gamma}{\rho+\gamma}}\right)^{\frac{\rho+\gamma}{\gamma}}$

$\forall s, \tilde{b}_s^{(l)} := \left(\frac{p}{K^\top a_s^{(l)}}\right)^{\frac{\rho}{\rho+\gamma}}$

$\forall s, b_s^{(l)} := \left(b_s^{(l-1)}\right)^{\tau} \left(\tilde{b}_s^{(l)}\right)^{1-\tau}$

od

Outputs: $P^{(L)}(D, \lambda) := p$

Appendix E. Additional results.

E.1. MNIST and Wasserstein geodesics. This subsection contains additional figures for the application of subsection 5.1.

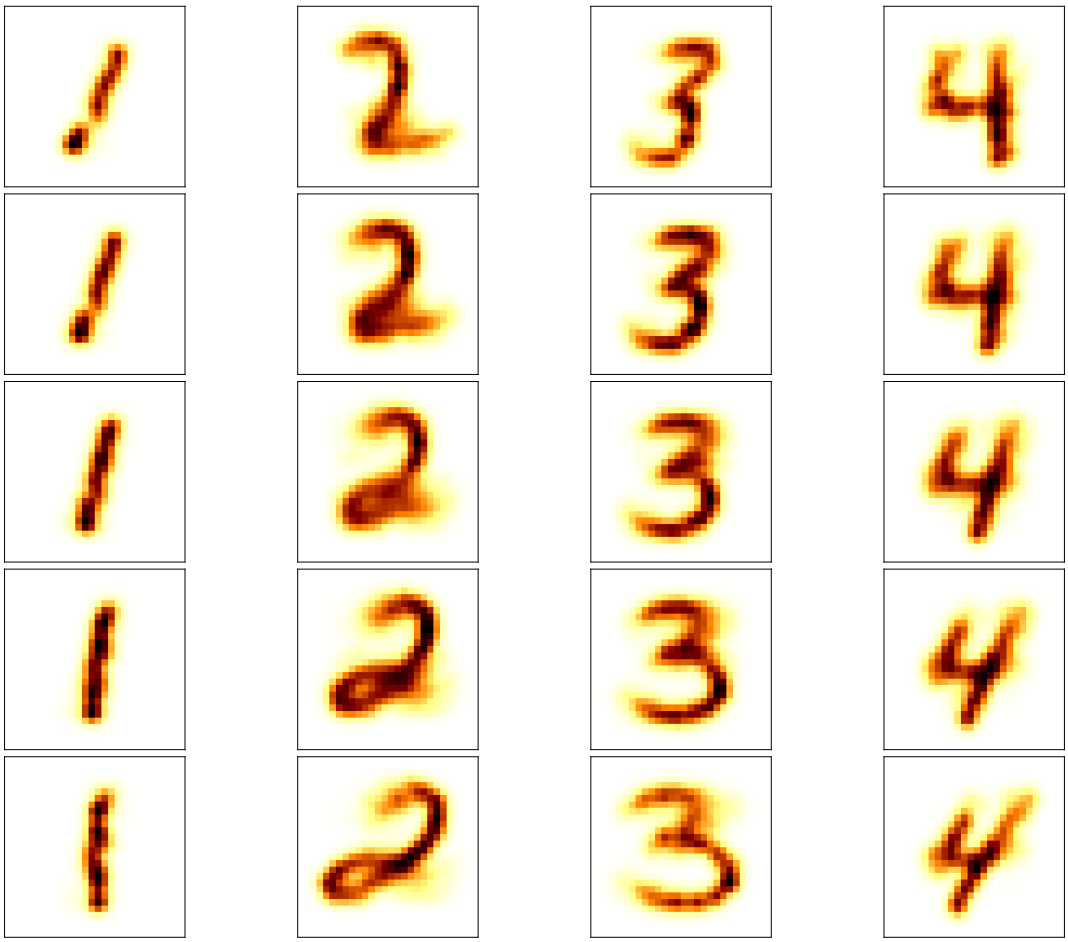


Figure E.1: Span of our two-atom dictionary for weights $(1-t,t), t \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$ when trained on images of digits $1, 2, 3, 4$. See the first columns of [74, Figure 5] for comparison with first WPGs.

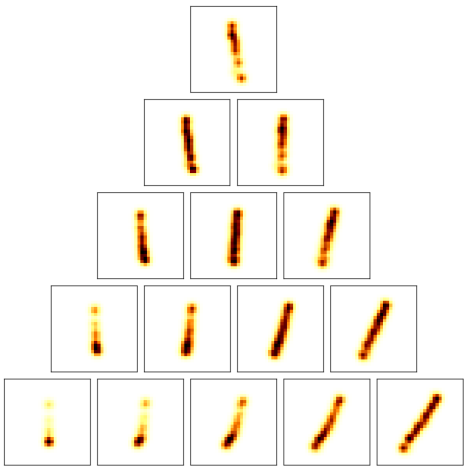


Figure E.2: Same as Figure 6 when training on images of the digit 1.

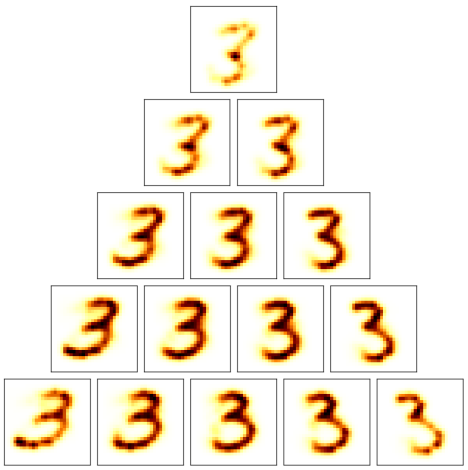


Figure E.3: Same as Figure 6 when training on images of the digit 3.

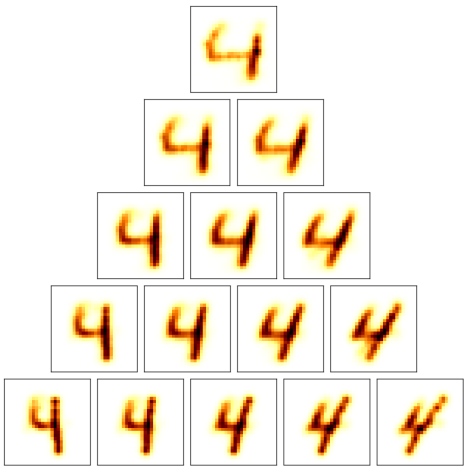


Figure E.4: Same as Figure 6 when training on images of the digit 4.

E.2. Point spread functions. This subsection contains additional figures for the application of subsection 5.2.

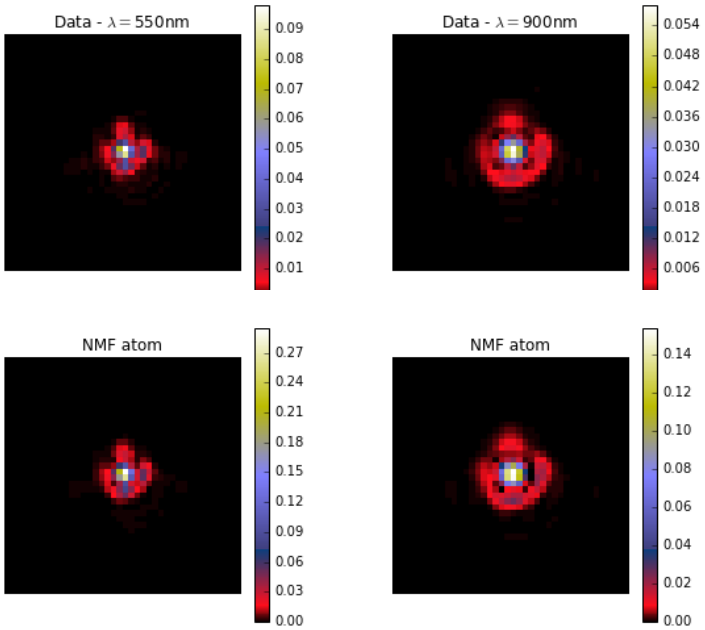


Figure E.5: Extreme wavelength PSFs in the dataset and atoms learned from NMF. See Figure 9 for those learned using our method.

E.3. Wasserstein faces. This subsection contains additional figures for the application of subsection 5.4.

Figure E.6: Similarly to Figure 14, we compare the atoms obtained using different loss functions, ranking them by mean PSNR: (a) $\overline{PSNR} = 33.81$, (b) $\overline{PSNR} = 33.72$, (c) $\overline{PSNR} = 32.95$, and (d) $\overline{PSNR} = 32.34$.

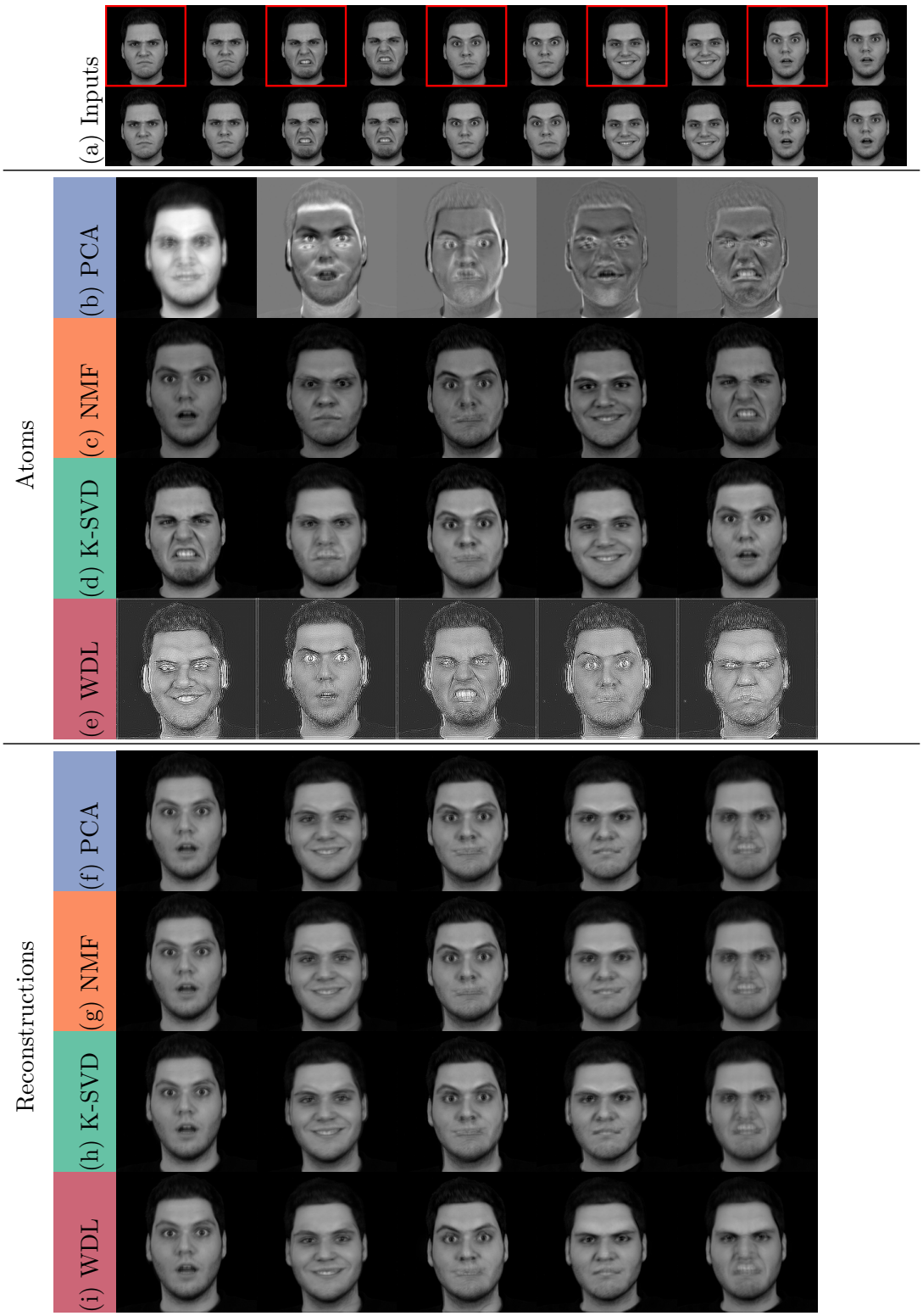


Figure E.7: Similarly to Figure 13, we compare our method to the Eigenfaces [81] approach, NMF and K-SVD as a tool to represent faces on a low-dimensional space.

Acknowledgements. This work was supported by the Centre National d'Etudes Spatiales (CNES), the European Community through the grants DEDALE (contract no. 665044) and NORIA (contract no. 724175) within the H2020 Framework Program and the French National Research Agency (ANR) through the grant ROOT. The authors are grateful to the anonymous referees for their salient comments and suggestions, as well as to the SIAM editorial team for their many suggested corrections.

REFERENCES

[1] M. Agueh and G. Carlier, *Barycenters in the wasserstein space*, SIAM Journal on Mathematical Analysis, 43 (2011), pp. 904–924.
[2] M. Aharon, M. Elad, and A. Bruckstein, *K-svd: An algorithm for designing overcomplete dictionaries for sparse representation*, IEEE Transactions on signal processing, 54 (2006), pp. 4311–4322.
[3] N. Aifanti, C. Papachristou, and A. Delopoulos, *The MUG facial expression database*, in Image Analysis for Multimedia Interactive Services (WIAMIS), 2010 11th International Workshop on, IEEE, 2010, pp. 1–4.
[4] J. Altschuler, J. Weed, and P. Rigollet, *Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration*, arXiv preprint arXiv:1705.09634, (2017).
[5] M. Arjovsky, S. Chintala, and L. Bottou, *Wasserstein GAN*. arXiv:1701.07875, 2017.
[6] F. Bassetti, A. Bodini, and E. Regazzini, *On minimum kantorovich distance estimators*, Statistics & probability letters, 76 (2006), pp. 1298–1302.
[7] F. Bassetti and E. Regazzini, *Asymptotic properties and robustness of minimum dissimilarity estimators of location-scale parameters*, Theory of Probability & Its Applications, 50 (2006), pp. 171–186.
[8] J.-D. Benamou, G. Carlier, M. Cuturi, L. Nenna, and G. Peyré, *Iterative bregman projections for regularized transportation problems*, SIAM Journal on Scientific Computing, 37 (2015), pp. A1111–A1138.
[9] E. Bernton, P. E. Jacob, M. Gerber, and C. P. Robert, *Inference in generative models using the wasserstein distance*, arXiv preprint arXiv:1701.05146, (2017).
[10] D. P. Bertsekas, *The auction algorithm: A distributed relaxation method for the assignment problem*, Annals of operations research, 14 (1988), pp. 105–123.
[11] J. Bigot, R. Gouet, T. Klein, and A. López, *Geodesic pca in the wasserstein space*, arXiv preprint arXiv:1307.7721, (2013).
[12] D. Blei and J. Lafferty, *Topic models*, Text mining: classification, clustering, and applications, 10 (2009), p. 71.
[13] E. Boissard, T. Le Gouic, J.-M. Loubes, et al., *Distribution template estimate with Wasserstein metrics*, Bernoulli, 21 (2015), pp. 740–759.
[14] N. Bonneel, G. Peyré, and M. Cuturi, *Wasserstein barycentric coordinates: Histogram regression using optimal transport*, ACM Transactions on Graphics (Proceedings of SIGGRAPH 2016), 35 (2016).
[15] N. Bonneel, J. Rabin, G. Peyré, and H. Pfister, *Sliced and Radon Wasserstein Barycenters of Measures*, Journal of Mathematical Imaging and Vision, 51 (2015), pp. 22–45.
[16] G. Carlier, A. Oberman, and E. Oudet, *Numerical methods for matching for teams and Wasserstein barycenters*, ESAIM: Mathematical Modelling and Numerical Analysis, (2015). to appear.
[17] L. Chizat, G. Peyré, B. Schmitzer, and F.-X. Vialard, *Scaling algorithms for unbalanced transport problems*, arXiv preprint arXiv:1607.05816, (2016).
[18] M. Cuturi, *Sinkhorn distances: Lightspeed computation of optimal transport*, in Advances in Neural Information Processing Systems, 2013, pp. 2292–2300.
[19] M. Cuturi and A. Doucet, *Fast computation of wasserstein barycenters*, in Proceedings of The 31st International Conference on Machine Learning, 2014, pp. 685–693.
[20] M. Cuturi and G. Peyré, *A smoothed dual approach for variational wasserstein problems*, SIAM Journal on Imaging Sciences, 9 (2016), pp. 320–343.
[21] A. d'Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. Lanckriet, *A direct formulation for sparse pca using semidefinite programming.*, SIAM review, 49 (2007), pp. 434–448.

[22] W. E. Deming and F. F. Stephan, *On a least squares adjustment of a sampled frequency table when the expected marginal totals are known*, The Annals of Mathematical Statistics, 11 (1940), pp. 427–444.
[23] S. Erlander and N. F. Stewart, *The gravity model in transportation analysis: theory and extensions*, vol. 3, Vsp, 1990.
[24] P. T. Fletcher, C. Lu, S. M. Pizer, and S. Joshi, *Principal geodesic analysis for the study of nonlinear statistics of shape.*, IEEE Transactions on Medical Imaging, 23 (2004), pp. 995–1005.
[25] J. Franklin and J. Lorenz, *On the scaling of multidimensional matrices*, Linear Algebra and its applications, 114 (1989), pp. 717–735.
[26] M. Fréchet, *Les éléments aléatoires de nature quelconque dans un espace distancié*, Ann. Inst. H. Poincaré, 10 (1948), pp. 215–310.
[27] C. Frogner, C. Zhang, H. Mobahi, M. Araya, and T. A. Poggio, *Learning with a wasserstein loss*, in Advances in Neural Information Processing Systems, 2015, pp. 2053–2061.
[28] W. Gao, J. Chen, C. Richard, and J. Huang, *Online dictionary learning for kernel lms*, IEEE Transactions on Signal Processing, 62 (2014), pp. 2765–2777.
[29] A. Genevay, G. Peyré, and M. Cuturi, *Learning generative models with sinkhorn divergences*, arXiv preprint arXiv:1706.00292, (2017).
[30] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, SIAM, 2008.
[31] S. Haker, L. Zhu, A. Tannenbaum, and S. Angenent, *Optimal mass transport for registration and warping*, International Journal of Computer Vision, 60 (2004), pp. 225–240.
[32] M. Harandi and M. Salzmann, *Riemannian coding and dictionary learning: Kernels to the rescue*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3926–3935.
[33] M. Harandi, C. Sanderson, C. Shen, and B. C. Lovell, *Dictionary learning and sparse coding on grassmann manifolds: An extrinsic solution*, in Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 3120–3127.
[34] M. T. Harandi, C. Sanderson, R. Hartley, and B. C. Lovell, *Sparse coding and dictionary learning for symmetric positive definite matrices: A kernel approach*, in Computer Vision–ECCV 2012, Springer, 2012, pp. 216–229.
[35] G. E. Hinton and R. R. Salakhutdinov, *Reducing the dimensionality of data with neural networks*, science, 313 (2006), pp. 504–507.
[36] J. Ho, Y. Xie, and B. Vemuri, *On a nonlinear generalization of sparse coding and dictionary learning*, in International conference on machine learning, 2013, pp. 1480–1488.
[37] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent component analysis*, vol. 46, John Wiley & Sons, 2004.
[38] Z. Irace and H. Batatia, *Motion-based interpolation to estimate spatially variant psf in positron emission tomography*, in Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European, IEEE, 2013, pp. 1–5.
[39] H. W. Kuhn, *The hungarian method for the assignment problem*, Naval research logistics quarterly, 2 (1955), pp. 83–97.
[40] R. Laureijs, J. Amiaux, S. Arduini, J.-L. Augueres, J. Brinchmann, R. Cole, M. Cropper, C. Dabin, L. Duvet, A. Ealet, et al., *Euclid definition study report*, arXiv preprint arXiv:1110.3193, (2011).
[41] D. D. Lee and H. S. Seung, *Learning the parts of objects by non-negative matrix factorization*, Nature, 401 (1999), pp. 788–791.
[42] H. Lee, A. Battle, R. Raina, and A. Y. Ng, *Efficient sparse coding algorithms*, in Advances in neural information processing systems, 2007, pp. 801–808.
[43] C. Léonard, *A survey of the Schrödinger problem and some of its connections with optimal transport*, Discrete and Continuous Dynamical Systems - Series A (DCDS-A), 34 (2014), pp. 1533–1574.
[44] P. Li, Q. Wang, W. Zuo, and L. Zhang, *Log-euclidean kernels for sparse representation and dictionary learning*, in Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 1601–1608.
[45] H. Liu, J. Qin, H. Cheng, and F. Sun, *Robust kernel dictionary learning using a whole sequence convergent algorithm.*, in IJCAI, vol. 1, 2015, p. 5.
[46] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, *Online learning for matrix factorization and sparse*

coding, Journal of Machine Learning Research, 11 (2010), pp. 19–60.
[47] S. Mallat, *A wavelet tour of signal processing*, Academic press, 1999.
[48] R. J. McCann, *A convexity principle for interacting gases*, Advances in mathematics, 128 (1997), pp. 153–179.
[49] Q. Mérigot, *A Multiscale Approach to Optimal Transport*, Computer Graphics Forum, (2011).
[50] G. Monge, *Mémoire sur la théorie des déblais et des remblais*, Histoire de l'Académie Royale des Sciences de Paris, (1781).
[51] G. Montavon, K.-R. Müller, and M. Cuturi, *Wasserstein training of restricted boltzmann machines*, in Advances in Neural Information Processing Systems, 2016, pp. 3711–3719.
[52] J. L. Morales and J. Nocedal, *Remark on "algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound constrained optimization"*, ACM Transactions on Mathematical Software (TOMS), 38 (2011), p. 7.
[53] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*, vol. 87, Springer Science & Business Media, 2013.
[54] F. Ngolè and J.-L. Starck, *Psf field learning based on optimal transport distances*, arXiv preprint arXiv:1703.06066, (2017).
[55] F. Ngolè, J.-L. Starck, S. Ronayette, K. Okumura, and J. Amiaux, *Super-resolution method using sparse regularization for point-spread function recovery*, Astronomy & Astrophysics, 575 (2015), p. A86.
[56] N. Papadakis, *Optimal Transport for Image Processing*, habilitation à diriger des recherches, Université de Bordeaux, Dec. 2015.
[57] K. Pearson, *Liii. on lines and planes of closest fit to systems of points in space*, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2 (1901), pp. 559–572.
[58] J. Pennington, R. Socher, and C. D. Manning, *Glove: Global Vectors for Word Representation.*, in EMNLP, vol. 14, 2014, pp. 1532–1543.
[59] G. Peyré, L. Chizat, F.-X. Vialard, and J. Solomon, *Quantum optimal transport for tensor field processing*, arXiv preprint arXiv:1612.08731, (2016).
[60] F. Pitié, A. C. Kokaram, and R. Dahyot, *N-dimensional probablility density function transfer and its application to colour transfer*, in Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2, ICCV '05, Washington, DC, USA, 2005, IEEE Computer Society, pp. 1434–1439.
[61] B. T. Polyak, *Some methods of speeding up the convergence of iteration methods*, USSR Computational Mathematics and Mathematical Physics, 4 (1964), pp. 1–17.
[62] Y. Quan, C. Bao, and H. Ji, *Equiangular kernel dictionary learning with applications to dynamic texture analysis*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 308–316.
[63] J. Rabin, G. Peyré, J. Delon, and M. Bernot, *Wasserstein barycenter and its application to texture mixing*, in International Conference on Scale Space and Variational Methods in Computer Vision, Springer, 2011, pp. 435–446.
[64] S. Rachev and L. Rüschendorf, *Mass Transportation Problems: Theory*, vol. 1, Springer Verlag, 1998.
[65] A. Rolet, M. Cuturi, and G. Peyré, *Fast dictionary learning with a smoothed wasserstein loss*, in Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, 2016, pp. 630–638.
[66] R. Rubinstein, M. Zibulevsky, and M. Elad, *Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit*, Cs Technion, 40 (2008), pp. 1–15.
[67] Y. Rubner, C. Tomasi, and L. J. Guibas, *The earth mover's distance as a metric for image retrieval*, Int. J. Comput. Vision, 40 (2000), pp. 99–121.
[68] G. Salton and M. J. McGill, *Introduction to modern information retrieval*, (1986).
[69] R. Sandler and M. Lindenbaum, *Nonnegative matrix factorization with earth mover's distance metric*, in Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 1873–1880.
[70] M. A. Schmitz, M. Heitz, N. Bonneel, F. Ngolè, D. Coeurjolly, M. Cuturi, G. Peyré, and J.-L. Starck, *Optimal transport-based dictionary learning and its application to euclid-like point spread function representation*, in SPIE Optical Engineering+ Applications, International Society for Optics and Photonics, 2017.

[71] B. Schmitzer, *Stabilized sparse scaling algorithms for entropy regularized transport problems*, arXiv preprint arXiv:1610.06519, (2016).
[72] B. Schölkopf, A. Smola, and K.-R. Müller, *Kernel principal component analysis*, Artificial Neural Networks — ICANN'97, (1997), pp. 583–588.
[73] E. Schrödinger, *Über die umkehrung der naturgesetze*, Verlag Akademie der wissenschaften in kommission bei Walter de Gruyter u. Company, 1931.
[74] V. Seguy and M. Cuturi, *Principal geodesic analysis for probability measures under the optimal transport metric*, in Advances in Neural Information Processing Systems, 2015, pp. 3312–3320.
[75] S. Shirdhonkar and D. W. Jacobs, *Approximate earth mover's distance in linear time*, in Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–8.
[76] R. Sinkhorn, *Diagonal equivalence to matrices with prescribed row and column sums*, The American Mathematical Monthly, 74 (1967), pp. 402–405.
[77] J. Solomon, F. De Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas, *Convolutional wasserstein distances: Efficient optimal transportation on geometric domains*, ACM Transactions on Graphics (TOG), 34 (2015), p. 66.
[78] J. Solomon, R. Rustamov, L. Guibas, and A. Butscher, *Wasserstein propagation for semi-supervised learning*, in Proceedings of The 31st International Conference on Machine Learning, 2014, pp. 306–314.
[79] M. Talagrand, *Transportation cost for gaussian and other product measures*, Geometric and Functional Analysis, 6 (1996), pp. 587–600.
[80] Theano Development Team, *Theano: A Python framework for fast computation of mathematical expressions*, arXiv e-prints, abs/1605.02688 (2016).
[81] M. Turk and A. Pentland, *Eigenfaces for Recognition*, Journal of Cognitive Neuroscience, 3 (1991), pp. 71–86.
[82] H. Van Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, *Design of non-linear kernel dictionaries for object recognition*, IEEE Transactions on Image Processing, 22 (2013), pp. 5123–5135.
[83] C. Villani, *Topics in optimal transportation*, no. 58, American Mathematical Soc., 2003.
[84] ———, *Optimal transport: old and new*, vol. 338, Springer Science & Business Media, 2008.
[85] W. Wang, D. Slepcev, S. Basu, J. A. Ozolek, and G. K. Rohde, *A linear optimal transportation framework for quantifying and visualizing variations in sets of images*, International Journal of Computer Vision, 101 (2013), pp. 254–269.
[86] J. Ye, P. Wu, J. Z. Wang, and J. Li, *Fast discrete distribution clustering using Wasserstein barycenter with sparse support*, IEEE Transactions on Signal Processing, 65 (2017), pp. 2317–2332.
[87] S. Zavriev and F. Kostyuk, *Heavy-ball method in nonconvex optimization problems*, Computational Mathematics and Modeling, 4 (1993), pp. 336–341.