On Transportation of Mini-batches: A Hierarchical Approach

Khai Nguyen [1] **Dang Nguyen** [2] **Quoc Nguyen** [2] **Tung Pham** [2] **Hung Bui** [2] **Dinh Phung** [3] **Trung Le** [3] **Nhat Ho** [1]

Abstract

Mini-batch optimal transport (m-OT) has been successfully used in practical applications that involve probability measures with a very high number of supports. The m-OT solves several smaller optimal transport problems and then returns the average of their costs and transportation plans. Despite its scalability advantage, the m-OT does not consider the relationship between mini-batches which leads to undesirable estimation. Moreover, the m-OT does not approximate a proper metric between probability measures since the identity property is not satisfied. To address these problems, we propose a novel mini-batch scheme for optimal transport, named *Batch of Mini-batches Optimal Transport* (BoMb-OT), that finds the optimal coupling between mini-batches and it can be seen as an approximation to a well-defined distance on the space of probability measures. Furthermore, we show that the m-OT is a limit of the entropic regularized version of the BoMb-OT when the regularized parameter goes to infinity. Finally, we carry out experiments on various applications including deep generative models, deep domain adaptation, approximate Bayesian computation, color transfer, and gradient flow to show that the BoMb-OT can be widely applied and performs well in various applications.

1. Introduction

Optimal transport (OT) (Villani, 2021; 2008; Peyré et al., 2019) has emerged as an efficient tool in dealing with problems involving probability measures. Under the name of Wasserstein distance, OT has been widely utilized to solve problems such as generative modeling (Arjovsky et al., 2017; Tolstikhin et al., 2018; Salimans et al., 2018; Genevay et al., 2018; Liutkus et al., 2019), barycenter problem (Ho et al., 2017; Li et al., 2020), and approximate Bayesian computation (Bernton et al., 2019a; Nadjahi et al., 2020). Furthermore, OT can also provide the most economical map of moving masses between probability measures, which is very useful in various tasks such as color transfer (Ferradans et al., 2014; Perrot et al., 2016), natural language processing (Alvarez-Melis & Jaakkola, 2018), domain adaptation (alignment) (Courty et al., 2016; Lee et al., 2019a; Xu et al., 2020), and graph processing (Titouan et al., 2019; Xu et al., 2019; Chen et al., 2020a).

Although OT has attracted growing attention in recent years, a major barrier that prevents OT from being ubiquitous is its heavy computational cost. When the two probability measures are discrete with n supports, solving the Wasserstein distance via the interior point methods has the complexity of $\mathcal{O}(n^3 \log n)$ (Pele & Werman, 2009), which is extremely expensive when n is large. There are two main lines of works that focus on easing this computational burden. The first approach is to find a good enough approximation of the solution by adding an entropic regularized term on the objective function (Cuturi, 2013). Several works (Altschuler et al., 2017; Lin et al., 2019) show that the entropic approach can produce a ε-approximated solution at the same time reduce the computational complexity to $\mathcal{O}(n^2/\varepsilon^2)$. The second line of works named the "slicing" approach is based on the closed-form of Wasserstein distance in one-dimensional space, which has the computational complexity of $\mathcal{O}(n \log n)$. There are various variants in this direction; i.e., (Bonneel et al., 2015; Deshpande et al., 2019; Kolouri et al., 2019; Nguyen et al., 2021a;b), these all belong to the family of sliced Wasserstein distances. Recently, some methods are proposed to combine the dimensional reduction approach with the entropic solver of Wasserstein distance to inherit advantages from both directions (Paty & Cuturi, 2019; Muzellec & Cuturi, 2019; Lin et al., 2021; 2020).

Although those works have reduced the computational cost of OT considerably, computing OT is nearly impossible in the big data settings where n could be as large as a few million. In particular, solving OT requires computing and storing a $n \times n$ cost matrix that is impractical with current computational devices. Especially in deep learning applications, both supports of empirical measures and the cost matrix must be stored in the same device (e.g. a GPU) for automatic differentiation. This problem exists in all variants

[1]Department of Statistics and Data Sciences, The University of Texas at Austin [2]VinAI Research [3]Monash University. Correspondence to: Khai Nguyen <khainb@utexas.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

of OT including Wasserstein distance, entropic Wasserstein, and sliced Wasserstein distance. Therefore, it leads to the development of the mini-batch methods for OT (Genevay et al., 2018; Sommerfeld et al., 2019), which we refer to as mini-batch OT loss. The main idea of the mini-batch method is to divide the original samples into multiple subsets (mini-batches), in the hope that each pair of subsets (mini-batches) could capture some structures of the two probability measures, meanwhile, computing OT cost between two mini-batches is cheap due to a very small size of mini-batches. Then the overall loss is defined as the average of distances between pairs of mini-batches. This scheme was applied for many forms of Wasserstein distances (Deshpande et al., 2018; Bhushan Damodaran et al., 2018; Kolouri et al., 2018; Salimans et al., 2018), and was theoretically studied in the works of (Bellemare et al., 2017; Bernton et al., 2019b; Nadjahi et al., 2019). Recently, Fatras et al. (Fatras et al., 2020; 2021b) formulated this approach by giving a formal definition of the mini-batch OT loss, studying its asymptotic behavior, and investigating its gradient estimation properties. Despite being applied successfully, the current mini-batch OT loss does not consider the relationship between mini-batches and treats every pair of mini-batches the same. This causes undesirable effects in measuring the discrepancy between probability measures. First, the m-OT loss is shown to be an approximation of a discrepancy (the population m-OT) that does not preserve the metricity property, namely, this discrepancy is always positive even when two probability measures are identical. Second, it is also unclear whether this discrepancy achieves the minimum value when the two probability measures are the same. That naturally raises the question of whether we could propose a better mini-batch scheme to sort out these issues to improve the performance of the OT in applications.

Contribution: In this paper, we propose a novel mini-batch scheme for optimal transport, which is named as *Batch of Mini-batches Optimal Transport* (BoMb-OT). In particular, the BoMb-OT views every mini-batch as a point in the product space, then a set of mini-batches could be considered as an empirical measure. We now could employ the Kantorovich formulation between these two empirical measures in the product space as a discrepancy between two sets of mini-batches. In summary, our main contributions are three-fold:

1. First, the BoMb-OT could provide a more similar transportation plan to the original OT than the m-OT, which leads to a more meaningful discrepancy using mini-batches. In particular, we prove that the BoMb-OT approximates a well-defined metric on the space of probability measures, named population BoMb-OT. Furthermore, the entropic regularization version of population BoMb-OT could be employed as a generalized version of the population m-OT. Specifically, when the regularization parameter in the entropic population BoMb-OT goes to infinity, its value approaches the value of the population m-OT.

2. Second, we present the implementation strategy of the BoMb-OT and detailed algorithms in various applications in Appendix C. We then demonstrate the favorable performance of the BoMb-OT over the m-OT in two main applications that use optimal transport losses, namely, deep generative models and deep domain adaptation. Moreover, we also compare BoMb-OT to m-OT in other applications, such as sample matching, approximate Bayesian computation, color transfer, and gradient flow. In all applications, we also provide a careful investigation of the effects of two hyper-parameters of the mini-batch scheme, which are the number of mini-batches and the size of mini-batches, on the performance of the BoMb-OT and the m-OT.

3. Third, we demonstrate that the idea of BoMb-OT, namely, the hierarchical approach can be applied to any type of transportation between mini-batch measures. As an instance, we extend m-UOT (Fatras et al., 2021a) to BoMb-UOT which improves the classification accuracy on target domains in deep domain adaptations on many standard datasets.

Organization: The remainder of the paper is organized as follows. In Section 2, we provide backgrounds for optimal transport distances and the conventional mini-batch scheme (m-OT). In Section 3, we define the new mini-batch scheme for optimal transport, Batch of mini-batches Optimal Transport, and derive some of its theoretical properties. Section 4 benchmarks the proposed mini-batch scheme by extensive experiments on large-scale datasets and followed by discussions in Section 5. Finally, proofs of key results and extra materials are in the supplementary.

Notation: For any probability measure μ on the Polish measurable space $(\mathcal{X}, \Sigma)$, we denote $\overset{\otimes m}{\mu}(m \geq 2)$ as the product measure on the product measurable space $(\mathcal{X}^m, \Sigma^m)$. For any $p \geq 1$, we define $\mathcal{P}_p(\mathbb{R}^N)$ as the set of Borel probability measures with finite p-th moment defined on a given metric space $(\mathbb{R}^N, \|.\|)$. To simplify the presentation, we abuse the notation by using both the notation X^m for both the random vector $(x_1, \ldots, x_m) \in \mathcal{X}^m$ and the set $\{x_1, \ldots, x_m\}$, and we define by $P_{X^m} := \frac{1}{m}\sum_{i=1}^m \delta_{x_i}$ the empirical measure (the mini-batch measure) associated with X^m. For any set $X^n := \{x_1, \ldots, x_n\}$ and $m \geq 1$, we denote by $[X^n]^m$ the product set of X^n taken m times and $\binom{X^n}{m}$ the set of all m-element subsets of X^n.

2. Background on mini-batch optimal transport

In this section, we first review the definitions of the Wasserstein distance, the entropic Wasserstein, and the sliced Wasserstein. We then review the definition of the mini-batch optimal transport (m-OT).

2.1. Wasserstein distance and its variants

We first start with the definition of Wasserstein distance and its variants. Let μ and ν be two probability measures on $\mathcal{P}_p(\mathbb{R}^N)$. The Wasserstein p-distance between μ and ν is defined as follows:

$$W_p(\mu,\nu) := \min_{\pi\in\Pi(\mu,\nu)} \left[\mathbb{E}_{\pi(x,y)}\|x-y\|^p\right]^{\frac{1}{p}}$$

, where $\Pi(\mu,\nu) := \left\{\pi : \int \pi dx = \nu, \int \pi dy = \mu\right\}$ is the set of transportation plans between μ and ν.

The entropic regularized Wasserstein to approximate the OT solution (Altschuler et al., 2017; Lin et al., 2019) between μ and ν is defined as follows (Cuturi, 2013): $W_p^\epsilon(\mu,\nu) := \min_{\pi\in\Pi(\mu,\nu)} \left\{\left[\mathbb{E}_{\pi(x,y)}\|x-y\|^p\right]^{\frac{1}{p}} + \epsilon \mathrm{KL}(\pi|\mu\otimes\nu)\right\}$, where $\epsilon > 0$ is a chosen regularized parameter and KL denotes the Kullback-Leibler divergence.

Finally, the sliced Wasserstein (SW) (Bonnotte, 2013; Bonneel et al., 2015) is motivated by the closed-form of the Wasserstein distance in one-dimensional space. The formal definition of sliced Wasserstein distance between μ and ν is: $SW_p(\mu,\nu) := \left[\mathbb{E}_{\theta\sim\mathcal{U}(\mathbb{S}^{N-1})} W_p^p(\theta\sharp\mu,\theta\sharp\nu)\right]^{\frac{1}{p}}$, where $\mathcal{U}(\mathbb{S}^{N-1})$ denotes the uniform measure over the $(N-1)$-dimensional unit hypersphere and $\theta\sharp$ is the orthogonal projection operator on direction θ.

2.2. Mini-batch Optimal Transport

In this section, we first discuss the memory issue of large-scale optimal transport and the challenges of the dual solver. Then, we revisit the mini-batch OT loss that has been used in training deep generative models, domain adaptations, color transfer, and approximate Bayesian computation (Bhushan Damodaran et al., 2018; Genevay et al., 2018; Tolstikhin et al., 2018; Fatras et al., 2020; Bernton et al., 2019a). To ease the presentation, we are given $X^n := \left\{x_1,\ldots,x_n\right\}$, $Y^n := \left\{y_1,\ldots,y_n\right\}$ i.i.d. samples from μ and ν in turn. Let $\mu_n := \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ and $\nu_n := \frac{1}{n}\sum_{i=1}^n \delta_{y_i}$ be two corresponding empirical measures from the whole data set. Here, n is usually large (e.g., millions), and each support in X^n, Y^n can be a high dimensional data point (e.g. a high-resolution image, video, etc).

Memory issue of optimal transport: Using an OT loss between μ_n and ν_n needs to compute and store a $n\times n$ cost matrix which has one trillion float entries when n is about millions. Moreover, when dealing with deep neural networks, both support points and the cost matrix are required to be stored in the same memory (e.g., a GPU with 8 gigabytes memory) as a part of the computational graph for automatic differentiation. This issue applies to all variants of OT losses, such as Wasserstein distance, entropic Wasserstein, sliced Wasserstein distance. Therefore, it is *nearly impossible* to compute OT and its variants in large-scale applications.

Challenges of stochastic dual solver: Using stochastic optimization to solve the Kantorovich dual form is a possible approach to deal with large-scale OT, i.e. Wasserstein GAN (Arjovsky et al., 2017; Leygonie et al., 2019). However, the obtained distance has been shown to be very different from the original Wasserstein distance (Mallasto et al., 2019; Stanczuk et al., 2021). Using input convex neural networks is another choice to approximate the Brenier potential (Makkuva et al., 2020). Nevertheless, recent work (Korotin et al., 2021) has indicated that input convex neural networks are not sufficient (have limited power) in approximating the Brenier potential. Furthermore, both mentioned approaches are restricted in the choice of ground metric. In particular, the $\mathcal{L}_1$ norm is used in Wasserstein GAN to make the constraint of dual form into the Lipchitz constraint and the $\mathcal{L}_2$ norm is for the existence of the Brenier potential.

Mini-batch solution: As a popular alternative approach for stable large-scale OT with flexible choices of the ground metric, the mini-batch optimal transport is proposed (Genevay et al., 2018; Sommerfeld et al., 2019) and has been widely used in various applications including generative modeling (Arjovsky et al., 2017; Deshpande et al., 2018; Salimans et al., 2018; Kolouri et al., 2018; Bunne et al., 2019), domain adaptation (Bhushan Damodaran et al., 2018; Lee et al., 2019a), and crowd counting (Wang et al., 2020). In this approach, the original n samples are divided into subsets (mini-batches) of size m, where m is often the largest number that the computer can process, then an alternative solution of the original OT problem is formed by aggregating these smaller OT solutions from mini-batches. In practice, mini-batch OT is often used in iterative procedures where each step requires transporting only a small fraction of samples e.g., estimating gradient of neural networks for stochastic optimization schemes.

We now state an adapted definition of the mini-batch OT (m-OT) scheme that was formulated in (Fatras et al., 2020), including its two key parts: its transportation cost and its transportation plan.

Definition 2.1 (Empirical m-OT)**.** For $p \geq 1$ and integers $m \geq 1$, and $k \geq 1$, let $d : \mathcal{P}_p(\mathcal{X}) \times \mathcal{P}_p(\mathcal{X}) \to [0,\infty)$ be a function, i.e., $\{W_p, W_p^\epsilon, SW_p\}$. Then, the mini-batch OT (m-OT) loss and the transportation plan, a $n\times n$ matrix, between μ_n and ν_n are defined as follows:

$$\widehat{U}_d^{k,m}(\mu_n,\nu_n) := \frac{1}{k^2}\sum_{i=1}^k\sum_{j=1}^k d(P_{X_i^m}, P_{Y_j^m});$$
$$\widehat{\pi}_k^m(\mu_n,\nu_n) := \frac{1}{k^2}\sum_{i=1}^k\sum_{j=1}^k \pi_{X_i^m,Y_j^m}, \quad (1)$$

where X_i^m is sampled with replacement from $\binom{X^n}{m}$, Y_i^m

is sampled with replacement from $\binom{Y^n}{m}$, and the transport plan $\pi_{X_i^m, Y_j^m}$, where its entries equal zero except those indexed of samples $X_i^m \times Y_j^m$, is the transportation plan when $d(P_{X_i^m}, P_{Y_j^m})$ is an optimal transport metric.

Compared to original definition in (Fatras et al., 2020), we consider k^2 pairs of mini-batches instead of k pairs. The above definition was generalized to the two original measures as follows in (Fatras et al., 2020):

Definition 2.2 (Population m-OT)**.** Assume that μ and ν are two probability measures on $\mathcal{P}_p(\mathcal{X})$ for given positive integers $p \geq 1$, $m \geq 1$, and $d : \mathcal{P}_p(\mathcal{X}) \times \mathcal{P}_p(\mathcal{X}) \to [0, \infty)$ be a given function. Then, the population mini-batch OT (m-OT) discrepancy between μ and ν is defined as follows:

$$U_d^m(\mu, \nu) := \mathbb{E}_{(X^m, Y^m) \sim \mu^{\otimes m} \otimes \nu^{\otimes m}} d(P_{X^m}, P_{Y^m}). \quad (2)$$

Issues of the m-OT: From Definition 2.1, the m-OT treats every pair of mini-batches the same by taking the average of the m-OT loss between any pair of them for both transportation loss and transportation plan. This treatment has some issues. First, a mini-batch is a sparse representation of the true distribution and two sparse representations of the same distribution could be very different from each other. Hence, a mini-batch from X^m would prefer to match to certain mini-batches of Y^m, rather than treating every mini-batch of Y^m equally. For example, each mini-batch has one datum, then each term in the population m-OT now is the ground metric of the OT cost, the population m-OT degenerates to $\mathbb{E}[d(X^m, Y^m)]$ for independent X^m and Y^m. This treatment also leads to an uninformative transportation plan shown in Figure 14, which is followed by a less meaningful transportation cost. Second, although it has been proved that the population m-OT is symmetric and positive (Fatras et al., 2020), for the same reason it does not vanish when two measures are identical.

3. Batch of Mini-batches Optimal Transport

To address the issues of m-OT, in this section, we propose a novel mini-batch scheme, named *batch of mini-batches OT (BoMb-OT)*. We first demonstrate the improvement of the BoMb-OT to the m-OT and discuss the practical usage of the BoMb-OT. Then, we prove that the BoMb-OT loss is an approximation of a well-defined metric in the space of Borel probability measures. Finally, we discuss the entropic version of the population BoMb-OT which admits the population m-OT as a special case where the entropic regularization goes to infinity in Appendix B. To simplify the presentation, we assume throughout this section that μ and ν are continuous probability measures in $\mathcal{P}_p(\mathcal{X})$ for some given $p \geq 1$ and $\mathcal{X} \subset \mathbb{R}^N$ where $N \geq 1$. Furthermore, $d : \mathcal{P}_p(\mathcal{X}) \times \mathcal{P}_p(\mathcal{X}) \to [0, \infty)$ is a given divergence between probability measures in $\mathcal{P}_p(\mathcal{X})$.

3.1. Definition of BoMb-OT and its properties

Intuitively, a mini-batch scheme in OT can be decomposed into two steps: (1) Solving local OT between every pair of mini-batches from samples of two original measures, (2) Combining local OT solutions into a global solution. As discussed above, the main problem with the m-OT lies in the second step, all pairs of mini-batches play the same role in the total transportation loss, meanwhile, a mini-batch from X^n would prefer to match some certain mini-batches from Y^n. To address the limitation, we enhance the second step by considering the optimal coupling between mini-batches. By doing so, we could define a new mini-batch scheme that is able to construct a good global mapping between samples that also leads to a meaningful objective loss.

Let $X_1^m, X_2^m, \ldots, X_k^m$ be mini-batches that are sampled with replacement from $[X^n]^m$ and $Y_1^m, Y_2^m, \ldots, Y_k^m$ be mini-batches that are sampled with replacement from $[Y^n]^m$. The batch of mini-batches optimal transport is defined as follows:

Definition 3.1 (Batch of mini-batches)**.** Assume that $p \geq 1$, $m \geq 1$, $k \geq 1$ are positive integers and let $d : \mathcal{P}_p(\mathcal{X}) \times \mathcal{P}_p(\mathcal{X}) \to [0, \infty)$ be a given function (e.g., $\{W_p, W_p^\epsilon, SW_p\}$). The *BoMb-OT loss* and the *BoMb-OT transportation plan* between probability measures μ_n and ν_n are given by:

$$\widehat{\mathcal{D}}_d^{k,m}(\mu_n, \nu_n) := \min_{\gamma \in \Pi(\overset{\otimes m}{\mu_k}, \overset{\otimes m}{\nu_k})} \sum_{i=1}^k \sum_{j=1}^k \gamma_{ij} d(P_{X_i^m}, P_{Y_j^m});$$
$$\widehat{\pi}_k^m(\mu_n, \nu_n) := \sum_{i=1}^k \sum_{j=1}^k \bar{\gamma}_{ij} \pi_{X_i^m, Y_j^m}, \quad (3)$$

where $\overset{\otimes m}{\mu_k} := \frac{1}{k} \sum_{i=1}^k \delta_{X_i^m}$ and $\overset{\otimes m}{\nu_k} := \frac{1}{k} \sum_{j=1}^k \delta_{Y_j^m}$ are two empirical measures defined on the product space via mini-batches (measures over mini-batches), $\bar{\gamma}$ is a $k \times k$ optimal transport plan between $\overset{\otimes m}{\mu_k}$ and $\overset{\otimes m}{\nu_k}$, and $\pi_{X_i^m, Y_j^m}$ is defined as in Definition 2.1.

Compared to the m-OT, the BoMb-OT considers an additional OT between two measures on mini-batches for combining mini-batch losses. The improvement of the BoMb-OT over the m-OT is illustrated in Figure 1 in which we assume that we can only solve 2×2 OT problems and we need to deal with the OT problems with two empirical measures of 3 supports $\mathcal{X}$ and $\mathcal{Y}$. The optimal transportation plan is to map i-th square (x_i) to i-th circle (y_i). Assume that there are four mini-batches: X_1^2 and X_2^2 from X^3 and Y_1^2 and Y_2^2 from Y^3. In m-OT, the weights for the pair of mini-batches X_1^2 and Y_2^2, X_2^2 and Y_1^2 equal $\frac{1}{4}$, meanwhile, the maps from X_1^2 to Y_2^2 and from X_2^2 to Y_1^2 are unuseful. In this toy example, X_1^2 should be mapped to Y_1^2 and X_2^2 is for Y_2^2. It is also the solution that the BoMb-OT tries to

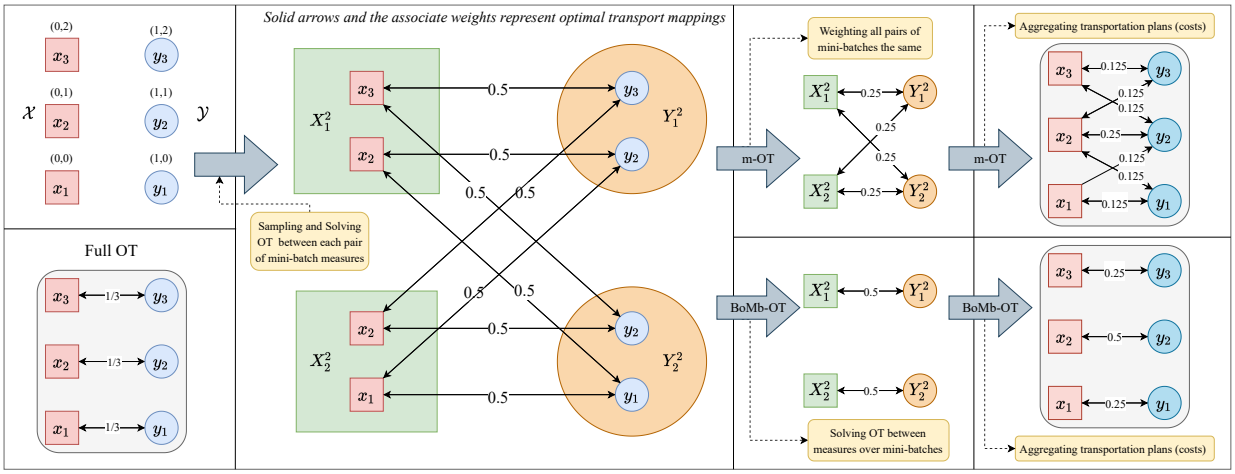

Figure 1. Visualization of the m-OT and the BoMb-OT in providing a mapping between samples.

obtain by re-weighting pairs of mini-batches through a transportation plan between two measures over mini-batches.

Practical usage of the BoMb-OT: There are three types of applications that the BoMb-OT can be utilized. The first one is *gradient-based* applications (deep learning) such as deep generative models, deep domain adaption, and gradient flow. In these applications, the goal is to estimate the gradient of a parameter of interest (a neural network) with an OT objective function. The second one is mapping-based applications, namely, we aim to obtain a transportation map between samples. Examples of mapping-based applications include color transfer, domain adaptation, and sample matching. Finally, the third type of application is *discrepancy-based* applications where we need to know the discrepancy between two measures, e.g., approximate Bayesian computation, and searching problems. We now discuss the implementation of BoMb-OT in each scenario.

Gradient-based (deep learning) applications: For a better presentation, we assume that a system with a GPU (graphics processing unit) is used in the training process. We further assume that the GPU has enough space to store the neural net, the $2 \times m$ supports of two mini-batch measures, and the $m \times m$ cost matrix. The algorithm of the BoMb-OT consists of three steps: *Forwarding*, *Solving $k \times k$ OT*, and *Re-forwarding and Backwarding*. In the first step, k^2 OT problems of size $m \times m$ are computed in turn on GPU to obtain the $k \times k$ cost matrix which indicates the transportation cost between each pair of mini-batch measures. We recall that it is only enough memory for computing $m \times m$ OT on GPU at a time. After that, an OT problem with the found $k \times k$ cost matrix is solved to find the coupling between mini-batches. Here, k can be much larger than m since the $k \times k$ OT can be solved on computer memory (RAM - Random-access memory), which is bigger than GPU's memory. We would like to emphasize that in the first two steps, automatic differentiation is not required. In the last step, we do re-forwarding and do backwarding (back-propagation) to obtain the gradients of each pair of mini-batches, then using the coupling in the second step to aggregate them into the final gradient signal of the parameter of interest. The detailed algorithms of the BoMb-OT for deep generative model and deep domain adaption are respectively described in Algorithm 1 in Appendix C.1 and Algorithm 2 in Appendix C.2. Compared to the m-OT, the BoMb-OT only costs an additional forwarding and an additional $k \times k$ OT problem that is not expensive since they do not require to create and store any computational graphs. In the case of training with multiple GPUs, we do not need the re-forwarding step.

Mapping-based applications: In this type of application, the BoMb-OT also consists of three steps like in gradient-based applications. However, in the last step, the transportation plan (the mapping) is aggregated instead of the gradient. As an example, we present the BoMb-OT algorithm for color transfer in Algorithm 3 in Appendix C.3. We would like to emphasize that the last step (re-forwarding) can be removed by storing all k^2 mini-batch transportation plans of size $m \times m$ which can be represented by a sparse matrix of size $n \times n$ for saving memory.

Discrepancy-based applications: The BoMb-OT needs only two steps in this scenario, namely, forwarding and solving $k \times k$ OT. Since we do not need to re-estimate any statistics from mini-batches, the final value of the BoMb-OT can be evaluated at the end of the second step without the re-forwarding step. Similar to the previous two types of applications, we present the BoMb-OT algorithm for approximate Bayesian computation in Algorithm 5 in Appendix C.4. Compared to the m-OT, the BoMb-OT needs only an additional $k \times k$ OT here.

Choosing k and m: In practice, we want to have as large as possible values of k and m. The mini-batch size m is often chosen to be the largest value of the computational memory that can be stored. For choosing k, there are two practical

cases. *Two (multiples) computational memories* - this is the case of modern computational systems that have at least a CPU (central processing unit) with RAM and a GPU with its corresponding memory. As discussed in the practical usage of the BoMb-OT, OT problems between mini-batch measures of size $m \times m$ are solved on the GPU for high computational speed in estimating the parameter of interest or other statistics. On the other hand, the $k \times k$ OT problems between measures over mini-batches can be computed on the CPU's memory. So, k can be chosen larger than m. *One (shared) computational memory* - in this case, the largest value of k equals m. Note that, smaller values of k and m can still be used for faster computation. We would like to recall that k, m are usually set to be relatively small to n (e.g., $k = 1$, $m = 100$) in recent applications (Genevay et al., 2018; Bhushan Damodaran et al., 2018; Kolouri et al., 2018; Bernton et al., 2019a). The reason is that the training process is iterative, and each iteration requires a stochastic estimation of the parameter (e.g., the gradient of the neural net), so there is no need to transport all n data samples.

Computational complexity of the BoMb-OT: The computational complexity of BoMb-OT depends on a particular choice of divergence d. In the experiments, we specifically consider three choices of $d \in \{W_p, W_p^\epsilon, SW_p\}$ where $\epsilon > 0$ is a chosen regularized parameter. Here, we only discuss the case when $d = W_p$, the discussion of other choices of d is in Appendix B. When d is the entropic Wasserstein distance, for each pair X_i^m and Y_j^m, the computational complexity of approximating $d(P_{X_i^m}, P_{Y_j^m})$ via the Sinkhorn algorithm is of order $\mathcal{O}(m^2/\varepsilon^2)$ where $\varepsilon > 0$ is some desired accuracy (Lin et al., 2019). Hence, the computation cost of k^2 OT transport plans from k^2 pairs of mini-batches is of order $\mathcal{O}(k^2 m^2/\varepsilon^2)$ when using entropic regularization (see Appendix B for the definition). With another OT cost within k^2 pairs, the total computational complexity of computing the BoMb-OT is $\mathcal{O}(k^2(m^2+1)/\varepsilon^2)$. It means that the computational complexity of computing the BoMb-OT is comparable to the m-OT, which is $\mathcal{O}(k^2 m^2/\varepsilon^2)$.

The BoMb-OT's transportation plan and cost: We discuss in detail the sparsity of BoMb-OT's transportation plan in Appendix B. To compare the BoMb-OT's plan with the m-OT's plan and the full-OT's plan, we carry out a simple experiment on two measures of 10 supports in Figure 14. We also compare the transportation cost of the BoMb-OT with the cost of m-OT and the cost of full-OT on two measures of 1000 supports in Figure 13. The details of the experiments and discussion are given in Appendix D.5. From this example, the BoMb-OT provides a similar plan and cost to the full-OT than the m-OT with various choices of m and k. We would like to recall that the full-OT cannot be used in real applications due to the impracticality of storing the full cost matrix between supports and high-dimensional supports as discussed in the Section 2.2.

Extension to BoMb-UOT: By changing d to unbalanced optimal transport (UOT) (Chizat et al., 2018), authors in (Fatras et al., 2021a) derived an extension of m-OT which is mini-batch unbalanced optimal transport (m-UOT). Similarly, we are able to extend BoMb-OT to BoMb-UOT. We present the definition of BoMb-UOT including the loss and the transportation plan in Definition B.1 in Appendix B.

3.2. Metric approximation of the BoMb-OT loss

In this section, we show that the BoMb-OT is an approximation of a well-defined metric, named *population BoMb-OT*, in the space of probability measures. To ease the ensuing discussion, we first define that metric as follows:

Definition 3.2 (Population BoMb-OT)**.** Let μ and ν be two probability measures on $\mathcal{P}_p(\mathcal{X})$, for $p \geq 1$ is a positive number. The *population BoMb-OT* between probability measures μ and ν is defined as:

$$\mathcal{D}_d^m(\mu,\nu) := \inf_{\gamma \in \Pi(\mu^{\otimes m}, \nu^{\otimes m})} \mathbb{E}_{(X^m,Y^m)\sim\gamma}[d(P_{X^m}, P_{Y^m})].$$

Different from the m-OT in Definition 2.2, the optimal plan between two distributions on product spaces is crucial to guarantee that the population BoMb-OT is a well-defined metric in the space of probability measures.

Theorem 3.3 (Metric property of population BoMb-OT)**.** *Assume that d is an invariant metric under permutation on $\mathcal{X}^m$ and the function $d(P_{X^m}, P_{Y^m})$ is continuous in terms of $X^m, Y^m \in \mathcal{X}^m$. Then, the population BoMb-OT is a well-defined metric in the space of probability measures.*

The proof of Theorem 3.3 is in Appendix A. The assumption of Theorem 3.3 is mild and satisfied when d is (sliced) Wasserstein metrics of order p, i.e., $d \in \{W_p, SW_p\}$.

Our next result shows the approximation error between the BoMb-OT loss and the population BoMb-OT distance. We provide the following result with the approximation error when $d \in \{W_p, SW_p\}$.

Theorem 3.4 (Population BoMb-OT)**.** *Assume that $p \geq 1$, $\mathcal{X}$ is a compact subset of $\mathbb{R}^N$, and all the possible mini-batches are considered. Then, we have (i) $\left|\widehat{\mathcal{D}}_d^{k,m}(\mu_n,\nu_n) - \mathcal{D}_d^m(\mu,\nu)\right| = \mathcal{O}_P\left(m^{1-\frac{1}{p}}/n^{\frac{1}{N}}\right)$ when $d \equiv W_p$; (ii) $\left|\widehat{\mathcal{D}}_d^{k,m}(\mu_n,\nu_n) - \mathcal{D}_d^m(\mu,\nu)\right| = \mathcal{O}_P\left(m^{1-\frac{1}{p}}/n^{\frac{1}{2}}\right)$ when $d \equiv SW_p$ and k is the number of mini-batches.*

The proof of Theorem 3.4 is in Appendix A. We would like to remark that the dependency of the sample complexity of the BoMb-OT on N is necessary when $d \equiv W_p$. It is due to the inclusion of the additional optimal transport in the BoMb-OT. On the other hand, the curse of dimensionality of the BoMb-OT loss does not happen when $d \equiv SW_p$. Furthermore, the choice $d \equiv SW_p$ improves not only the

Table 1. Comparison between the BoMb-OT and the m-OT on deep generative models. On the MNIST dataset, we evaluate the performances of generators by computing approximated Wasserstein-2 while we use the FID score (Heusel et al., 2017) on CIFAR10 and CelebA. The qualitative results (randomly generated images) are given in Figures 4-6 in Appendix D.1.

Dataset	k	m-OT(W_2^ϵ)	BoMb-OT(W_2^ϵ)	m-OT(SW_2)	BoMb-OT(SW_2)
MNIST	1	28.12	28.12	37.57	37.57
	2	27.88	**27.53**	36.01	**35.27**
	4	27.60	**27.41**	35.18	**34.19**
	8	27.36	**27.10**	34.33	**33.17**
CIFAR10	1	78.34	78.34	80.51	80.51
	2	76.20	**74.25**	67.86	**62.80**
	4	76.01	**74.12**	62.30	**58.78**
	8	75.22	**73.33**	59.68	**53.44**
CelebA	1	54.16	54.16	90.33	90.33
	2	52.85	**51.53**	82.45	**74.48**
	4	52.56	**50.55**	73.06	**72.19**
	8	51.92	**49.63**	71.95	**68.52**

Table 2. Comparison between two mini-batch schemes on the deep domain adaptation on digits datasets. We varied the number of mini-batches k and reported the classification accuracy on the target domain. Experiments were run three times.

Scenario	k	m-OT	BoMb-OT	m-UOT	BoMb-UOT
S → M	1	90.98 ± 1.00	90.98 ± 1.00	99.10 ± 0.05	99.10 ± 0.05
	2	92.66 ± 0.71	$\mathbf{93.36 \pm 0.51}$	98.99 ± 0.08	$\mathbf{99.15 \pm 0.07}$
	4	93.70 ± 0.53	$\mathbf{94.79 \pm 0.18}$	99.08 ± 0.08	$\mathbf{99.19 \pm 0.05}$
	8	94.17 ± 0.39	$\mathbf{95.23 \pm 0.44}$	98.94 ± 0.03	$\mathbf{99.09 \pm 0.07}$
U → M	1	92.64 ± 0.45	92.64 ± 0.45	98.14 ± 0.21	98.14 ± 0.21
	2	92.85 ± 0.34	$\mathbf{94.76 \pm 0.12}$	98.34 ± 0.11	$\mathbf{98.44 \pm 0.05}$
	4	93.83 ± 0.56	$\mathbf{95.64 \pm 0.26}$	98.55 ± 0.04	$\mathbf{98.60 \pm 0.02}$
	8	94.69 ± 0.73	$\mathbf{95.90 \pm 0.33}$	98.62 ± 0.08	$\mathbf{98.70 \pm 0.06}$

sample complexity but also the computational complexity of the BoMb-OT. In particular, in Appendix B, we demonstrate that the computational complexity of the BoMb-OT is $\mathcal{O}(k^2 m \log m)$ when $d \equiv SW_p$. Another potential scenario that the BoMb-OT does not have the curse of dimensionality is when we consider its entropic regularized version in equation 5. The entropic optimal transport had been shown to have sample complexity at the order $n^{-\frac{1}{2}}$ (Mena & Weed, 2019). In the case of the entropic regularized BoMb-OT (see equation (5) in Appendix B), when the regularized parameter λ is infinity, namely, the m-OT, the result of (Fatras et al., 2020; 2021b) already established the sample complexity $n^{-\frac{1}{2}}$. However, for a general value of the regularized parameter, it is unclear whether we still have this sample complexity $n^{-\frac{1}{2}}$ of the entropic BoMb-OT. We leave this question for future works.

4. Experiments

In this section, we demonstrate the favorable performance of BoMb-OT compared to m-OT in three discussed types of applications, namely, *gradient-based, mapping-based, and value-based* applications. For gradient-based applications, we run experiments on deep generative model (Genevay et al., 2018; Deshpande et al., 2018), deep domain adaptation (Bhushan Damodaran et al., 2018). Experiments on color transfer (Rabin et al., 2014; Ferradans et al., 2014) are conducted as examples for mapping-based applications. Lastly, we present results on approximate Bayesian computation (Bernton et al., 2019a) which is a value-based application. In the main text of the paper, we only report and discuss the obtained experimental results, the details of applications and their algorithms are given in Appendix C. In Appendix D, we provide detailed experimental results of applications including visualization and computational time. Moreover, we also carry out experiments on gradient flow (Santambrogio, 2017) and visualize mini-batch transportation matrices of the m-OT and the BoMb-OT. From all experiments, we observe that the BoMb-OT performs better than m-OT consistently. The detailed settings of our experiments including neural network architecture, hyperparameters, and evaluation metrics are in Appendix E. [1]

Table 3. Comparison between two mini-batch schemes on the deep domain adaptation on the VisDA dataset. We varied the number of mini-batches k and reported the classification accuracy on the target domain. Experiments were run three times.

k	m-OT	BoMb-OT	m-UOT	BoMb-UOT
1	65.29 ± 0.26	65.29 ± 0.26	72.07 ± 0.07	72.07 ± 0.07
2	65.46 ± 0.33	$\mathbf{66.98 \pm 0.09}$	72.52 ± 0.14	$\mathbf{73.72 \pm 0.13}$
4	65.51 ± 0.17	$\mathbf{67.71 \pm 0.05}$	72.95 ± 0.06	$\mathbf{74.65 \pm 0.03}$

4.1. Deep generative model

We now show the deep generative model result on MNIST (LeCun et al., 1998), CIFAR10 (32x32) (Krizhevsky et al., 2009), and CelebA (64x64) (Liu et al., 2015) using different mini-batch schemes with two OT losses for d: SW_2 (Deshpande et al., 2018) and W_2^ϵ (Genevay et al., 2018). In this task, we update the generative neural networks by using the stochastic gradient that is calculated from the mini-batch losses with different values of the number of mini-batches k. The details of the application and algorithm are given in Appendix C.1. We set $m = 100$ for MNIST, $m = 50$ for CIFAR10 and CelebA, other hyperparameters and evaluation metrics are deferred to Appendix E.1. According to Table 1, the BoMb-OT always gives lower quantitative metrics than the m-OT on all datasets with all choices of the number of mini-batches k and mini-batch ground metric d. From the table, we also observe that increasing the number of mini-batches k improves the result of generators that are trained with the same number of stochastic gradient updates. It suggests that the gradient of an OT loss be improved by more than 1 pair of mini-batches like the way it has been implemented in practice. Since the application of

[1] Python code is published at https://github.com/UT-Austin-Data-Science-Group/Mini-batch-OT.

Table 4. Summary results of deep domain adaptation on Office-Home dataset. Experiments were run 3 times. (*) denotes the result taken from JUMBOT's paper (Fatras et al., 2021a).

Methods	A2C	A2P	A2R	C2A	C2P	C2R	P2A	P2C	P2R	R2A	R2C	R2P	Avg
RESNET-50 (*)	34.90	50.00	58.00	37.40	41.90	46.20	38.50	31.20	60.40	53.90	41.20	59.90	46.10
DANN (*)	44.30	59.80	69.80	48.00	58.30	63.00	49.70	42.70	70.60	64.00	51.70	78.30	58.30
CDAN-E (*)	52.50	71.40	76.10	59.70	69.90	71.50	58.70	50.30	77.50	70.50	57.90	83.50	66.60
ALDA (*)	52.20	69.30	76.40	58.70	68.20	71.10	57.40	49.60	76.80	70.60	57.30	82.50	65.80
ROT (*)	47.20	71.80	76.40	58.60	68.10	70.20	56.50	45.00	75.80	69.40	52.10	80.60	64.30
m-OT	49.76	68.37	74.40	59.64	64.69	68.63	56.12	46.69	74.37	67.27	54.45	77.95	63.53
m-UOT	54.99	74.45	**80.78**	65.66	**74.93**	74.91	64.70	**53.42**	80.01	**74.58**	59.88	**83.73**	70.17
BoMb-OT (Ours)	50.16	69.57	74.84	60.24	65.18	69.15	57.48	47.42	74.88	67.39	54.19	78.59	64.09
BoMb-UOT (Ours)	**56.23**	**75.24**	80.53	**65.80**	74.57	**75.38**	**66.15**	53.21	**80.03**	74.25	**60.12**	83.30	**70.40**

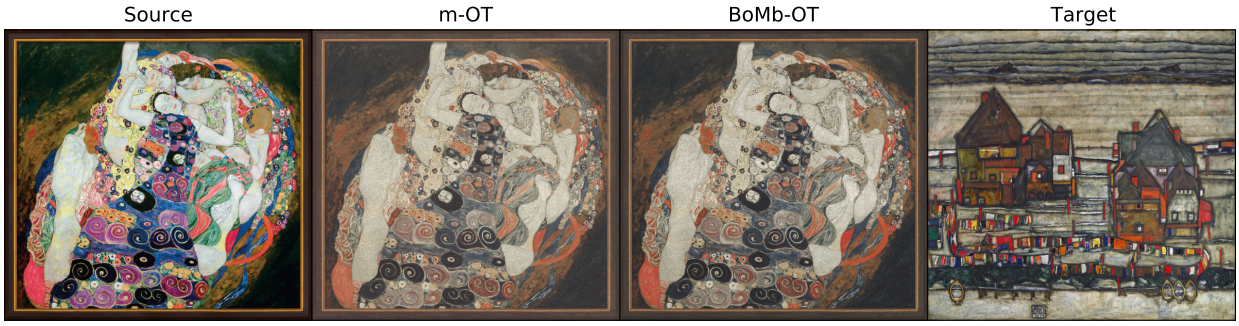

Figure 2. The transferred images of the m-OT and the BoMb-OT with k=10, m=10, and the full-OT from the most left source image to the most right target image.

UOT in deep generative model still remains a challenge, we are not able to compare the BoMb-UOT and the m-UOT in this task. So, we leave this challenge for future works.

4.2. Deep domain adaptation

In this section, we conduct deep domain adaptation on three datasets including digits, VisDA, and Office-Home. For digits datasets, we adapt two digits datasets SVHN (Netzer et al., 2011) and USPS (Hull, 1994) to MNIST (LeCun et al., 1998). We vary the number of mini-batches k and compare the performance of two mini-batch schemes. Similar to generative models, we train the neural networks for feature encoder and classes predictor by the stochastic gradients of mini-batch losses including m-OT, m-UOT, BoMb-OT, and BoMb-UOT. The details of the application and algorithm are given in Appendix C.2. The mini-batch size m is set to 50 on the SVHN dataset, 25 on the USPS dataset, 72 on the VisDA dataset, and 65 on the Office-Home dataset. Other detailed settings including architectures of neural networks and hyper-parameters settings are given in Appendix E.2. We would like to recall that we aim to compare our proposed hierarchical method and the conventional averaging method, not UOT and OT.

Table 2 illustrates the performance of different methods on digits datasets. Comparing between two schemes, the BoMb-OT and the BoMb-UOT always achieve better results than their counterparts for all choices of k on digits datasets. In terms of VisDA, the results are summarized in Table 3. As can be seen from the table, the hierarchical approaches, BoMb-(U)OT can increase the accuracy from 1% to 2%. Further more, we observe that the classification accuracy on the target domain generally improves as k increases but keeping the same number of stochastic gradient updates for deep neural networks.

On Office-Home dataset (Venkateswara et al., 2017), Table 4 summarizes the classification accuracy on the target domain. We compare our method against other DA methods: DANN (Ganin et al., 2016), CDAN-E (Long et al., 2018), m-OT (DeepJDOT) (Bhushan Damodaran et al., 2018), ALDA (Chen et al., 2020b), ROT (Balaji et al., 2020), and m-UOT (JUMBOT) (Fatras et al., 2021a). BoMb-UOT produces the highest performance in 7 out of 12 domain adaptation tasks, leading to a margin of 0.23 on average compared to other competitors. To show the effectiveness of our scheme when dealing with more than one mini-batch, we also conduct a similar experiment as on the VisDA dataset. Detailed results of changing the number of mini-batches is given in Table 7 in Appendix D.2.

4.3. Color Transfer

In this section, we show that our new mini-batch strategy can improve further the quality of the color transfer. The details

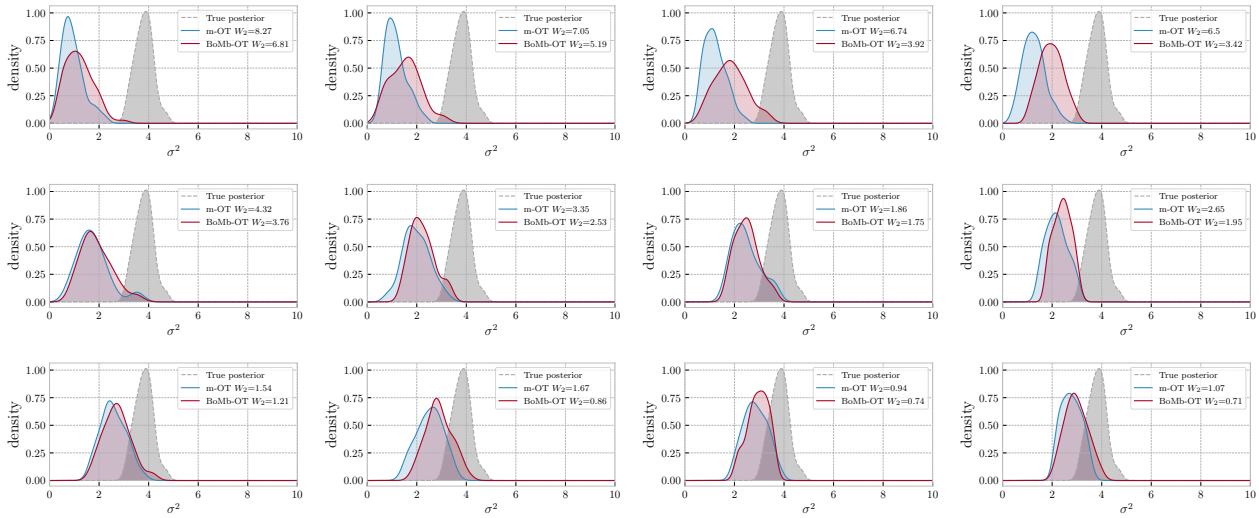

Figure 3. Approximated posteriors from ABC with the m-OT and the BoMb-OT. The first row, the second row, and the last row have $m = 8$, $m = 16$, and $m = 32$, respectively. In each row, the number of mini-batches k is 2, 4, 6, and 8 from left to right.

of the application and algorithm are given in Appendix C.3. Based on the experiment, we observe that the BoMb-OT provides a better barycentric mapping for color transfer than m-OT with every setting of k and m. Here, we show some transfer images in Figure 2. We can observe that the color of the BoMb-OT's image is more similar to the target image than one of the m-OT. The color palettes in Appendix D.3 also reinforce this claim. Moreover, we show that the transferred image of the BoMb-OT is closer to the full-OT's transferred image than the one of m-OT in Figure 9 in Appendix D.3. We would like to recall that the full-OT is not scalable in practice while mini-batch methods can transform color between two images that contain millions of pixels (Fatras et al., 2021b).

4.4. Approximate Bayesian Computation (ABC)

As choices of sample acceptance-rejection criteria, we compare the m-OT and the BoMb-OT in ABC. We present the detail of the application and the algorithm in Appendix C.4. In detail, we use the m-OT and the BoMb-OT with the Wasserstein-2 ground metric for the acceptance-rejection sampling's criteria in sequential Monte Carlo ABC (Toni et al., 2009) (using the implementation from pyABC (Klinger et al., 2018) with 100 particles and 10 iterations). We use the same setup as in (Nadjahi et al., 2020): there are $n = 100$ observations $\{x_i\}_{i=1}^n$ i.i.d from multivariate Gaussian $\mathcal{N}(\mu_*, \sigma_*^2 I_N)$ where N is the dimension, $\mu_* \sim \mathcal{N}(0, I_N)$ and $\sigma_*^2 = 4$. The task is to estimate the posterior of σ^2 under the imaginary assumption that σ^2 follows inverse gamma distribution $\mathcal{IG}(1, 1)$. Under these assumptions, the posterior of σ^2 has the form $\mathcal{IG}(1 + n\frac{d}{2}, 1 + \frac{1}{2}\sum_{i=1}^n ||x_i - \nu_*||^2)$.

After obtaining all samples, we estimate their densities by utilizing Gaussian kernel density estimation and then plot the approximated posteriors and the true posterior in Figure 3. Here we run the algorithm with several values of (k, m), namely, $m \in \{8, 16, 32\}$ and $k \in \{2, 4, 6, 8\}$. From these results, we see that a bigger m usually returns a better posterior in both the m-OT and the BoMb-OT cases. Similarly, increasing k also improves the performance of the two methods. Moreover, these graphs strengthen the claim that the BoMb-OT is better than the m-OT in every setting of (k, m) since its posteriors are always closer to the true posterior than those of the m-OT in both visual result and Wasserstein-2 distance.

5. Conclusion

In the paper, we have presented a novel mini-batch method for optimal transport, named Batch of Mini-batches Optimal Transport (BoMb-OT). The idea of the BoMb-OT is to consider the optimal transport problem on the space of mini-batches with an OT-types ground metric. We prove that the BoMb-OT is an approximation of a valid distance between probability measures and its entropic regularized version is the generalization of the conventional mini-batch optimal transport. More importantly, we have shown that the BoMb-OT can be implemented efficiently and they have more favorable performance than the m-OT in various applications of optimal transport including deep generative models, deep domain adaptation, color transfer, approximate Bayesian computation, and gradient flow. For future work, we could consider a hierarchical approach version of optimal transport between incomparable spaces.

Acknowledgement. A part of this work was partially done when Khai Nguyen worked at VinAI Research in 2021.

References

Altschuler, J., Niles-Weed, J., and Rigollet, P. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. In *Advances in neural information processing systems*, pp. 1964–1974, 2017.

Alvarez-Melis, D. and Jaakkola, T. Gromov-Wasserstein alignment of word embedding spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1881–1890, 2018.

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

Aude, G., Cuturi, M., Peyré, G., and Bach, F. Stochastic optimization for large-scale optimal transport. *Advances in Neural Information Processing Systems*, 29, 2016.

Balaji, Y., Chellappa, R., and Feizi, S. Robust optimal transport with applications in generative modeling and domain adaptation. *Advances in Neural Information Processing Systems*, 33, 2020.

Bellemare, M. G., Danihelka, I., Dabney, W., Mohamed, S., Lakshminarayanan, B., Hoyer, S., and Munos, R. The Cramer distance as a solution to biased Wasserstein gradients. *arXiv preprint arXiv:1705.10743*, 2017.

Berkes, I. and Philipp, W. An almost sure invariance principle for the empirical distribution function of mixing random variables. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 41(2):115–137, 1977.

Bernton, E., Jacob, P. E., Gerber, M., and Robert, C. P. Approximate Bayesian computation with the Wasserstein distance. *Journal of the Royal Statistical Society*, 2019a.

Bernton, E., Jacob, P. E., Gerber, M., and Robert, C. P. On parameter estimation with the Wasserstein distance. *Information and Inference: A Journal of the IMA*, 8(4): 657–676, 2019b.

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

Bobkov, S. and Ledoux, M. 'One-dimensional empirical measures, order statistics, and Kantorovich transport distances. *Memoirs of the American Mathematical Society*, 261, 2019.

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

Bonnotte, N. *Unidimensional and evolution methods for optimal transportation*. PhD thesis, Paris 11, 2013.

Bunne, C., Alvarez-Melis, D., Krause, A., and Jegelka, S. Learning generative models across incomparable spaces. In *International Conference on Machine Learning*, pp. 851–861, 2019.

Chen, L., Gan, Z., Cheng, Y., Li, L., Carin, L., and Liu, J. Graph optimal transport for cross-domain alignment. In *International Conference on Machine Learning*, pp. 1542–1553. PMLR, 2020a.

Chen, M., Zhao, S., Liu, H., and Cai, D. Adversarial-learned loss for domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3521–3528, 2020b.

Chizat, L., Peyré, G., Schmitzer, B., and Vialard, F.-X. Unbalanced optimal transport: Dynamic and Kantorovich formulations. *Journal of Functional Analysis*, 274(11): 3090–3123, 2018.

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

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pp. 2292–2300, 2013.

De Acosta, A. Invariance principles in probability for triangular arrays of b-valued random vectors and some applications. *The Annals of Probability*, pp. 346–373, 1982.

Deshpande, I., Zhang, Z., and Schwing, A. G. Generative modeling using the sliced Wasserstein distance. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3483–3491, 2018.

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

Dobrić, V. and Yukich, J. E. Asymptotics for transportation cost in high dimensions. *Journal of Theoretical Probability*, 8:97–118, 1995.

Dudley, R. M. The speed of mean Glivenko-Cantelli convergence. *Annals of Mathematical Statistics*, 40:40–50, 1969.

Fatras, K., Zine, Y., Flamary, R., Gribonval, R., and Courty, N. Learning with minibatch Wasserstein: asymptotic and gradient properties. In *AISTATS 2020-23nd International Conference on Artificial Intelligence and Statistics*, volume 108, pp. 1–20, 2020.

Fatras, K., Sejourne, T., Flamary, R., and Courty, N. Unbalanced minibatch optimal transport; applications to domain adaptation. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 3186–3197. PMLR, 18–24 Jul 2021a. URL http://proceedings.mlr.press/v139/fatras21a.html.

Fatras, K., Zine, Y., Majewski, S., Flamary, R., Gribonval, R., and Courty, N. Minibatch optimal transport distances; analysis and applications. *arXiv preprint arXiv:2101.01792*, 2021b.

Ferradans, S., Papadakis, N., Peyré, G., and Aujol, J.-F. Regularized discrete optimal transport. *SIAM Journal on Imaging Sciences*, 7(3):1853–1882, 2014.

Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouve, A., and Peyré, G. Interpolating between optimal transport and MMD using Sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2681–2690, 2019.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL http://jmlr.org/papers/v22/20-451.html.

Fournier, N. and Guillin, A. On the rate of convergence in Wasserstein distance of the empirical measure. *Probability Theory and Related Fields*, 162:707–738, 2015.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

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

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

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

Hull, J. J. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.

Klinger, E., Rickert, D., and Hasenauer, J. Pyabc: distributed, likelihood-free inference. *Bioinformatics*, 34 (20):3591–3593, 2018.

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

Kolouri, S., Nadjahi, K., Simsekli, U., Badeau, R., and Rohde, G. Generalized sliced Wasserstein distances. In *Advances in Neural Information Processing Systems*, pp. 261–272, 2019.

Korotin, A., Li, L., Genevay, A., Solomon, J., Filippov, A., and Burnaev, E. Do neural optimal transport solvers work? A continuous Wasserstein-2 benchmark. *Proceedings in Neural Information Processing Systems*, 34, 2021.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

Lee, J., Dabagia, M., Dyer, E., and Rozell, C. Hierarchical optimal transport for multimodal distribution alignment. In *Advances in Neural Information Processing Systems*, pp. 13474–13484, 2019b.

Leygonie, J., She, J., Almahairi, A., Rajeswar, S., and Courville, A. Adversarial computation of optimal transport maps. *arXiv preprint arXiv:1906.09691*, 2019.

Li, L., Genevay, A., Yurochkin, M., and Solomon, J. M. Continuous regularized Wasserstein barycenters. *Advances in Neural Information Processing Systems*, 33, 2020.

Lin, T., Ho, N., and Jordan, M. On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms. In *International Conference on Machine Learning*, pp. 3982–3991, 2019.

Lin, T., Zheng, Z., Chen, E. Y., Cuturi, M., and Jordan, M. I. On projection robust optimal transport: Sample complexity and model misspecification. *arXiv preprint arXiv:2006.12301*, 2020.

Lin, T., Zheng, Z., Chen, E., Cuturi, M., and Jordan, M. On projection robust optimal transport: Sample complexity and model misspecification. In *International Conference on Artificial Intelligence and Statistics*, pp. 262–270. PMLR, 2021.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Liutkus, A., Simsekli, U., Majewski, S., Durmus, A., and Stöter, F.-R. Sliced-Wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In *International Conference on Machine Learning*, pp. 4104–4113. PMLR, 2019.

Long, M., Cao, Z., Wang, J., and Jordan, M. I. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pp. 1645–1655, 2018.

Luo, D., Xu, H., and Carin, L. Hierarchical optimal transport for robust multi-view learning. *arXiv preprint arXiv:2006.03160*, 2020.

Makkuva, A., Taghvaei, A., Oh, S., and Lee, J. Optimal transport mapping via input convex neural networks. In *International Conference on Machine Learning*, pp. 6672–6681. PMLR, 2020.

Mallasto, A., Montúfar, G., and Gerolin, A. How well do WGANs estimate the Wasserstein metric? *arXiv preprint arXiv:1910.03875*, 2019.

Mena, G. and Weed, J. Statistical bounds for entropic optimal transport: sample complexity and the central limit theorem. In *Advances in Neural Information Processing Systems*, 2019.

Muzellec, B. and Cuturi, M. Subspace detours: Building transport plans that are optimal on subspace projections. In *Advances in Neural Information Processing Systems*, pp. 6917–6928, 2019.

Nadjahi, K., Durmus, A., Simsekli, U., and Badeau, R. Asymptotic guarantees for learning generative models with the sliced-Wasserstein distance. In *Advances in Neural Information Processing Systems*, pp. 250–260, 2019.

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

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Nguyen, K. and Ho, N. Amortized projection optimization for sliced Wasserstein generative models. *arXiv preprint arXiv:2203.13417*, 2022a.

Nguyen, K. and Ho, N. Revisiting sliced Wasserstein on images: From vectorization to convolution. *arXiv preprint arXiv:2204.01188*, 2022b.

Nguyen, K., Ho, N., Pham, T., and Bui, H. Distributional sliced-Wasserstein and applications to generative modeling. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=QYjO70ACDK.

Nguyen, K., Nguyen, S., Ho, N., Pham, T., and Bui, H. Improving relational regularized autoencoders with spherical sliced fused Gromov-Wasserstein. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=DiQD7FWL233.

Paty, F.-P. and Cuturi, M. Subspace robust Wasserstein distances. In *International Conference on Machine Learning*, pp. 5072–5081, 2019.

Pele, O. and Werman, M. Fast and robust earth mover's distance. In *ICCV*. IEEE, 2009.

Perrot, M., Courty, N., Flamary, R., and Habrard, A. Mapping estimation for discrete optimal transport. *Advances in Neural Information Processing Systems*, 29:4197–4205, 2016.

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

Rabin, J., Ferradans, S., and Papadakis, N. Adaptive color transfer with relaxed optimal transport. In *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 4852–4856. IEEE, 2014.

Salimans, T., Zhang, H., Radford, A., and Metaxas, D. Improving GANs using optimal transport. In *International Conference on Learning Representations*, 2018.

Santambrogio, F. {Euclidean, metric, and Wasserstein} Gradient Flows: An Overview. *Bulletin of Mathematical Sciences*, 7(1):87–154, 2017.

Sommerfeld, M., Schrieber, J., Zemel, Y., and Munk, A. Optimal transport: Fast probabilistic approximation with exact solvers. *Journal of Machine Learning Research*, 20 (105):1–23, 2019.

Stanczuk, J., Etmann, C., Kreusser, L. M., and Schönlieb, C.-B. Wasserstein GANs work because they fail (to approximate the Wasserstein distance). *arXiv preprint arXiv:2103.01678*, 2021.

Titouan, V., Courty, N., Tavenard, R., and Flamary, R. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*, pp. 6275–6284, 2019.

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

Toni, T., Welch, D., Strelkowa, N., Ipsen, A., and Stumpf, M. P. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31): 187–202, 2009.

Venkateswara, H., Eusebio, J., Chakraborty, S., and Panchanathan, S. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5018–5027, 2017.

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

Villani, C. *Topics in optimal transportation*, volume 58. American Mathematical Soc., 2021.

Wang, B., Liu, H., Samaras, D., and Hoai, M. Distribution matching for crowd counting. In *Advances in Neural Information Processing Systems*, 2020.

Xu, H., Luo, D., and Carin, L. Scalable Gromov-Wasserstein learning for graph partitioning and matching. In *Advances in neural information processing systems*, pp. 3052–3062, 2019.

Xu, R., Liu, P., Wang, L., Chen, C., and Wang, J. Reliable weighted optimal transport for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4394–4403, 2020.

Yurochkin, M., Claici, S., Chien, E., Mirzazadeh, F., and Solomon, J. M. Hierarchical optimal transport for document representation. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/8b5040a8a5baf3e0e67386c2e3a9b903-Paper.pdf.

Supplement to "On Transportation of Mini-batches: A Hierarchical Approach"

In this supplementary material, we collect several proofs and remaining materials that were deferred from the main paper. In Appendix A, we provide the proofs of the main results in the paper. We present additional materials including discussions about computation complexity of the BoMb-OT, the connection of the BoMb-OT to hierarchical optimal transport, entropic regularization of BoMb-OT, sparsity of the BoMb-OT, extension to BoMb-UOT, and extension with non-OT mini-batch metrics in Appendix B. Furthermore, we provide a description of applications and the BoMb-OT algorithms in those applications in Appendix C. Additional results of presented experiments in the main text in Appendix D, and their corresponding settings in Appendix E.

A. Proofs

In this appendix, we give detailed proofs of theorems that are stated in the main paper.

A.1. Proof of Theorem 3.3

We first prove that for any probability measures μ and $\nu \in \mathcal{P}_p(\Theta)$, there exists an optimal transportation plan γ^* such that

$$\mathcal{D}_d^m(\mu,\nu) = \mathbb{E}_{(X^m,Y^m)\sim\gamma^*}[d(P_{X^m},P_{Y^m})]. \tag{4}$$

From the definition of the BoMb-OT, there is a sequence of transportation plans $\gamma_n \in \Pi(\overset{\otimes m}{\mu},\overset{\otimes m}{\nu})$ such that

$$\mathbb{E}_{(X^m,Y^m)\sim\gamma_n}[d(P_{X^m},P_{Y^m})] \to \mathcal{D}_d(\mu,\nu)$$

as $n \to \infty$. Since $\Pi(\overset{\otimes m}{\mu},\overset{\otimes m}{\nu})$ is compact in the weak* topology (Villani, 2008), γ_n weakly converges to some $\gamma^* \in \Pi(\overset{\otimes m}{\mu},\overset{\otimes m}{\nu})$. Since $d(P_{x^m},P_{y^m})$ is continuous in terms of $x^m, y^m \in \mathcal{X}^m$, an application of Portmanteau theorem leads to

$$\lim_{n\to\infty} \mathbb{E}_{(X^m,Y^m)\sim\gamma_n}[d(P_{X^m},P_{Y^m})] \geq \mathbb{E}_{(X^m,Y^m)\sim\gamma^*}[d(P_{X^m},P_{Y^m})].$$

Putting the above results together, we obtain

$$\mathcal{D}_d^m(\mu,\nu) = \mathbb{E}_{(X^m,Y^m)\sim\gamma^*}[d(P_{X^m},P_{Y^m})].$$

Therefore, there exists an optimal transportation plan γ^* such that equation 4 holds.

We now proceed to prove that the BoMb-OT is a well-defined metric in the space of Borel probability measures. First, we demonstrate that $\mathcal{D}_d(\mu,\nu) = 0$ if and only if $\mu = \nu$. In fact, from the definition of $\mathcal{D}_d(.,.)$, if we have two probability measures μ and ν such that $\mathcal{D}(\mu,\nu) = 0$, we find that

$$\inf_{\gamma\in\Pi(\overset{\otimes m}{\mu},\overset{\otimes m}{\nu})} \mathbb{E}_{(X^m,Y^m)\sim\gamma}[d(P_{X^m},P_{Y^m})] = 0.$$

Since d is a metric, it implies that there exists transportation plan $\gamma^* \in \Pi(\overset{\otimes m}{\mu},\overset{\otimes m}{\nu})$ such that $P_{X^m} = P_{Y^m}$ γ^*-almost everywhere. It demonstrates that for each x, there exists a permutation σ_x of $\{1,\ldots,m\}$ such that $x_i = y_{\sigma_x(i)}$ for all $i \in [m]$. Now, for any test function $f : \mathcal{X} \to \mathbb{R}$, we have

$$\begin{aligned}
\int_{\mathcal{X}^m} \prod_{i=1}^m f(x_i) d\overset{\otimes m}{\mu}(x) = \int_{\mathcal{X}^m\times\mathcal{X}^m} \prod_{i=1}^m f(x_i) d\gamma^*(x,y) &= \int_{\mathcal{X}^m\times\mathcal{X}^m} \prod_{i=1}^m f(y_{\sigma_x(i)}) d\gamma^*(x,y) \\
&= \int_{\mathcal{X}^m\times\mathcal{X}^m} \prod_{i=1}^m f(y_i) d\gamma^*(x,y) \\
&= \int_{\mathcal{X}^m} \prod_{i=1}^m f(y_i) d\overset{\otimes m}{\nu}(y).
\end{aligned}$$

Since $\int_{\mathcal{X}^m} \prod_{i=1}^m f(x_i) d\overset{\otimes m}{\mu}(x) = \left(\int_{\mathcal{X}} f(x_1) d\mu(x_1)\right)^m$ and $\int_{\mathcal{X}^m} \prod_{i=1}^m f(y_i) d\overset{\otimes m}{\nu}(y) = \left(\int_{\mathcal{X}} f(y_1) d\nu(y_1)\right)^m$, the above results show that

$$\int_{\mathcal{X}} f(x_1) d\mu(x_1) = \int_{\mathcal{X}} f(y_1) d\nu(y_1)$$

for any test function $f : \mathcal{X} \to \mathbb{R}$. It indicates that $\mu = \nu$. On the other hand, if $\mu = \nu$, we have $\overset{\otimes m}{\mu} = \overset{\otimes m}{\nu}$. We can construct the transportation plan $\bar{\gamma}(x, y) = \delta_x(y)\overset{\otimes m}{\mu}(x)$ for all (x, y). Then, we find that

$$\mathcal{D}_d^m(\mu, \nu) \leq \mathbb{E}_{(X,Y)\sim\bar{\gamma}}[d(P_{X^m}, P_{Y^m})] = 0.$$

Therefore, we obtain the conclusion that $\mathcal{D}(\mu, \nu) = 0$ if and only if $\mu = \nu$.

Now, we demonstrate the triangle inequality property of $\mathcal{D}_d(.,.)$, namely, we will show that for any probability measures μ_1, μ_2, and μ_3, we have

$$\mathcal{D}_d^m(\mu_1, \mu_2) + \mathcal{D}_d^m(\mu_2, \mu_3) \geq \mathcal{D}_d^m(\mu_1, \mu_3).$$

The proof of this inequality is a direct application of gluing lemma (Berkes & Philipp, 1977; De Acosta, 1982). In particular, for any transportation plans $\gamma_1 \in \Pi(\overset{\otimes m}{\mu}_1, \overset{\otimes m}{\mu}_2)$ and $\gamma_2 \in \Pi(\overset{\otimes m}{\mu}_2, \overset{\otimes m}{\mu}_3)$, we can construct a probability measure ξ on $\mathcal{X}^m \times \mathcal{X}^m \times \mathcal{X}^m$ such that $\xi(.,., \mathcal{X}^m) = \gamma_1(.,.)$ and $\xi(\mathcal{X}^m, .,.) = \gamma_2(.,.)$. Therefore, we find that

$$\begin{aligned}
&\mathbb{E}_{(X,Y)\sim\gamma_1}[d(P_{X^m}, P_{Y^m})] + \mathbb{E}_{(Y,Z)\sim\gamma_2}[d(P_{Y^m}, P_{Z^m})] \\
&\quad= \int_{(\mathcal{X}^m)^3} \Big[d(P_{x^m}, P_{y^m}) + d(P_{y^m}, P_{z^m})\Big] d\xi(x, y, z) \\
&\quad\geq \int_{(\mathcal{X}^m)^3} d(P_{x^m}, P_{z^m}) d\xi(x, y, z) \geq \mathcal{D}_d^m(\mu_1, \mu_3).
\end{aligned}$$

Taking the infimum on the LHS of the above inequality with respect to γ_1 and γ_2, we obtain the triangle inequality property of $\mathcal{D}(.,.)$. As a consequence, we reach the conclusion of the theorem.

A.2. Proof of Theorem 3.4

To ease the presentation, for any product measures $\overset{\otimes m}{\mu}$ and $\overset{\otimes m}{\nu}$, we denote the following loss between $\overset{\otimes m}{\mu}$ and $\overset{\otimes m}{\nu}$ as follows:

$$\bar{\mathcal{D}}_d^m(\overset{\otimes m}{\mu}, \overset{\otimes m}{\nu}) := \inf_{\gamma \in \Pi(\overset{\otimes m}{\mu}, \overset{\otimes m}{\nu})} \mathbb{E}_{(X^m, Y^m)\sim\gamma}[d(P_{X^m}, P_{Y^m})].$$

From the definitions of the BoMb-OT losses, we have $\mathcal{D}_d^m(\mu, \nu) = \bar{\mathcal{D}}_d^m(\overset{\otimes m}{\mu}, \overset{\otimes m}{\nu})$ and $\widehat{\mathcal{D}}_d^{k,m}(\mu_n, \nu_n) = \bar{\mathcal{D}}_d^m(\overset{\otimes m}{\mu_k}, \overset{\otimes m}{\nu_k})$. Using the similar proof argument as that of Theorem 3.3, we can check that $\bar{\mathcal{D}}_d^m$ satisfies the triangle inequality, namely, for any product measures $\overset{\otimes m}{\mu}$, $\overset{\otimes m}{\nu}$, and $\overset{\otimes m}{\eta}$, we have $\bar{\mathcal{D}}_d^m(\overset{\otimes m}{\mu}, \overset{\otimes m}{\nu}) + \bar{\mathcal{D}}_d^m(\overset{\otimes m}{\nu}, \overset{\otimes m}{\eta}) \geq \bar{\mathcal{D}}_d^m(\overset{\otimes m}{\mu}, \overset{\otimes m}{\eta})$.

From the above definition and properties of $\bar{\mathcal{D}}$, we find that

$$\begin{aligned}
\left|\widehat{\mathcal{D}}_d^{k,m}(\mu_n, \nu_n) - \mathcal{D}_d^m(\mu, \nu)\right| &= \left|\bar{\mathcal{D}}_d^m(\overset{\otimes m}{\mu_k}, \overset{\otimes m}{\nu_k}) - \bar{\mathcal{D}}_d^m(\overset{\otimes m}{\mu}, \overset{\otimes m}{\nu})\right| \\
&\leq \left|\bar{\mathcal{D}}_d^m(\overset{\otimes m}{\mu_k}, \overset{\otimes m}{\nu_k}) - \bar{\mathcal{D}}_d^m(\overset{\otimes m}{\mu}, \overset{\otimes m}{\nu_k})\right| + \left|\bar{\mathcal{D}}_d^m(\overset{\otimes m}{\mu}, \overset{\otimes m}{\nu_k}) - \bar{\mathcal{D}}_d^m(\overset{\otimes m}{\mu}, \overset{\otimes m}{\nu})\right| \\
&\leq \bar{\mathcal{D}}_d^m(\overset{\otimes m}{\mu_k}, \overset{\otimes m}{\mu}) + \bar{\mathcal{D}}_d^m(\overset{\otimes m}{\nu_k}, \overset{\otimes m}{\nu}).
\end{aligned}$$

(i) Since $d \equiv W_p$ where $p \geq 1$, we have

$$\begin{aligned}
d(P_{X^m}, P_{Y^m}) = W_p(P_{X^m}, P_{Y^m}) &= \frac{1}{m^{1/p}} \left(\inf_{\sigma} \sum_{i=1}^m \|X_i - Y_{\sigma(i)}\|^p\right)^{1/p} \\
&\leq \frac{1}{m^{1/p}} \sum_{i=1}^m \|X_i - Y_i\|,
\end{aligned}$$

where the infimum is taken over all possible permutations σ of $\{1, 2, \ldots, m\}$. The above inequality indicates that

$$\bar{\mathcal{D}}_d^m(\overset{\otimes m}{\mu_k}, \overset{\otimes m}{\mu}) \leq \inf_{\gamma \in \Pi(\overset{\otimes m}{\mu_k}, \overset{\otimes m}{\mu})} \mathbb{E}_{(X^m, Y^m) \sim \gamma} \left[\frac{1}{m^{1/p}} \sum_{i=1}^{m} \|X_i - Y_i\| \right] = m^{1-1/p} \cdot W_1(\mu_n, \mu).$$

Since $\mathcal{X}$ is a compact subset of $\mathbb{R}^N$, the results of (Dudley, 1969; Dobrić & Yukich, 1995; Fournier & Guillin, 2015) show that $W_1(\mu_n, \mu) = \mathcal{O}_P(n^{-1/N})$. Collecting these results together, we have

$$\bar{\mathcal{D}}_d^m(\overset{\otimes m}{\mu_k}, \overset{\otimes m}{\mu}) = \mathcal{O}_P\left(\frac{m^{1-1/p}}{n^{1/N}}\right).$$

With similar argument, we also find that $\bar{\mathcal{D}}_d^m(\overset{\otimes m}{\nu_k}, \overset{\otimes m}{\nu}) = \mathcal{O}_P\left(\frac{m^{1-1/p}}{n^{1/N}}\right)$. Putting all the above results together, we reach the conclusion of part (i) of the theorem.

(ii) The proof of part (ii) follows directly from the argument of part (i) and the results that $SW_p(\mu_n, \mu) = \mathcal{O}_P(n^{-1/2})$ and $SW_p(\nu_n, \nu) = \mathcal{O}_P(n^{-1/2})$ when $\mathcal{X}$ is a compact subset of $\mathbb{R}^N$ (Bobkov & Ledoux, 2019). Therefore, we obtain the conclusion of part (ii) of the theorem.

B. Additional materials

Computational complexity: We now discuss the computational complexities of the BoMb-OT loss when $d \equiv \{W_p^\epsilon, SW_p\}$. When $d \equiv W_p^\epsilon$ for some given regularized parameter $\epsilon > 0$, we can use the Sinkhorn algorithm to compute $d(P_{X_i^m}, P_{Y_j^m})$ for any $1 \leq i, j \leq k$. The computational complexity of the Sinkhorn algorithm for computing it is of the order $\mathcal{O}(m^2)$. Therefore, the computational complexity of computing the cost matrix from the BoMb-OT is of the order $\mathcal{O}(m^2k^2)$. Given the cost matrix, the BoMb-OT loss can be approximated with the complexity of the order $\mathcal{O}(k^2/\varepsilon^2)$ where ε stands for the desired accuracy. As a consequence, the total computational complexity of computing the BoMb-OT is $\mathcal{O}(m^2k^2 + k^2/\varepsilon^2)$.

When $d \equiv SW_p$, the computational complexity for computing $d(P_{X_i^m}, P_{Y_j^m})$ is of the order $\mathcal{O}(m \log m)$ for any $1 \leq i, j \leq k$. It shows that the complexity of computing the cost matrix from the BoMb-OT loss is of the order $\mathcal{O}(m(\log m)k^2)$. Given the cost matrix, the complexity of approximating the BoMb-OT is at the order of $\mathcal{O}(k^2/\varepsilon^2)$. Hence, the total complexity is at the order of $\mathcal{O}(m(\log m)k^2 + k^2/\varepsilon^2)$.

Connection to hierarchical OT: At the first sight, the BoMb-OT may look similar to hierarchical optimal transport (HOT). However, we would like to specify the main difference between the BoMb-OT and the HOT. In particular, on the one hand, the HOT comes from the hierarchical structure of data. For example, Yurochkin et al. (Yurochkin et al., 2019) consider optimal transport problems on both the document level and the word level for document representation. In the paper (Luo et al., 2020), HOT is proposed to handle multi-view data which is collected from different sources. Similarly, a hierarchical formulation of OT is proposed in (Lee et al., 2019b) to leverage cluster structure in data to improve alignment. On the other hand, the BoMb-OT makes *no assumption* about the hierarchical structure of data. We consider the optimal coupling of mini-batches as we want to improve the quality of mini-batch loss and its transportation plan.

Entropic regularized population BoMb-OT: We now consider an entropic regularization of the population BoMb-OT, which is particularly useful for reducing the computational complexity of the BoMb-OT. In particular, the *entropic regularized population BoMb-OT* between two probability measures μ and ν admits the following form:

$$\mathcal{ED}_d^m(\mu, \nu) := \min_{\gamma \in \Pi(\overset{\otimes m}{\mu}, \overset{\otimes m}{\nu})} \mathbb{E}_{(X^m, Y^m) \sim \gamma}[d(P_{X^m}, P_{Y^m})] + \lambda \cdot \text{KL}(\gamma | \overset{\otimes m}{\mu} \otimes \overset{\otimes m}{\nu}), \tag{5}$$

where $\lambda > 0$ stands for a chosen regularized parameter. From the above definition, the entropic regularized population BoMb-OT is an interpolation between the population BoMb-OT and the population m-OT. On the one hand, when $\lambda \to 0$, we have $\mathcal{ED}_d(\mu, \nu)$ converges to $\mathcal{D}_d(\mu, \nu)$. On the other hand, when $\lambda \to \infty$, the joint distribution γ approaches $\overset{\otimes m}{\mu} \otimes \overset{\otimes m}{\nu}$ and $\mathcal{ED}_d(\mu, \nu)$ converges to $U_d(\mu, \nu)$. The transportation plan of the entropic regularized BoMb-OT can be derived by simply replacing the coupling between mini-batches in Definition 3.1 with the coupling found by the Sinkhorn algorithm. To ease the presentation, we will refer to the entropic regularized BoMb-OT as BoMb-OT with $\lambda > 0$.

Sparsity of transportation plan from BoMb-OT: We assume that we are dealing with two empirical measures of n supports. For $d = W_p$, the optimal transportation plan contains n entries that are not zero. In the m-OT case, with k

mini-batches of size m, the m-OT's transportation plan contains at most k^2m non-zero entries. On the other hand, with k mini-batches of size m, the BoMb-OT's transportation plan has at most km (which is often much smaller than n) non-zero entries. For $d = W_p^\epsilon$ ($\epsilon > 0$), the optimal transportation plans contains at most n^2 non-zero entries. In this case, the m-OT provides transportation plans that have at most k^2m^2 non-zero entries. The BoMb-OT's transportation plans contain at most km^2 non-zero entries. The sparsity is useful in the re-forward step of the BoMb-OT algorithm since we can skip pair of mini-batches that has zero mass to save computation.

Extension to BoMb-UOT: We first review the definition of unbalanced optimal transport (UOT). The unbalanced optimal transport (Chizat et al., 2018) between μ_n and ν_n is defined as follows:

$$\text{UOT}_\phi^\tau(\mu_n, \nu_n) := \min_{\pi \in \mathbb{R}_+^{n\times n}} \langle C, \pi\rangle + \tau \text{D}_\phi(\pi_1, \mu_n) + \tau \text{D}_\phi(\pi_2, \nu_n),$$

where C is the distance matrix e.g., $\mathcal{L}_2$, $\tau > 0$ is a regularized parameter, D_ϕ is a certain probability divergence (e.g., KL divergence), and π_1, π_2 are respectively the marginal distributions of non-negative measure π and correspond to μ_n, ν_n. We now consider the definition of the BoMb-UOT loss and the BoMb-UOT transportation plan.

Definition B.1 (BoMb-UOT)**.** Assume that $p \geq 1$, $m \geq 1$, $k \geq 1$ are positive integers. The *BoMb-UOT loss* and the *BoMb-UOT transportation plan* between probability measures μ_n and ν_n are given by:

$$\widehat{\mathcal{DU}}_{UOT}^{k,m}(\mu_n, \nu_n) := \min_{\gamma \in \Pi(\overset{\otimes m}{\mu_k}, \overset{\otimes m}{\nu_k})} \sum_{i=1}^k \sum_{j=1}^k \gamma_{ij} UOT_\phi^\tau(P_{X_i^m}, P_{Y_j^m});$$
$$\widehat{\pi}_{k,\phi}^{m,\tau}(\mu_n, \nu_n) := \sum_{i=1}^k \sum_{j=1}^k \gamma_{ij} \pi_{X_i^m, Y_j^m}^{\tau,\phi}, \tag{6}$$

where $\overset{\otimes m}{\mu_k} := \frac{1}{k}\sum_{i=1}^k \delta_{X_i^m}$ and $\overset{\otimes m}{\nu_k} := \frac{1}{k}\sum_{j=1}^k \delta_{Y_j^m}$ are two empirical measures defined on the product space via mini-batches (measures over mini-batches), $\bar{\gamma}$ is a $k \times k$ optimal transport plan between $\overset{\otimes m}{\mu_k}$ and $\overset{\otimes m}{\nu_k}$, and $\pi_{X_i^m, Y_j^m}^\tau$ is the incomplete mini-batch transportation plan from UOT.

Non-OT choice of d: We would like to recall that d in Definition 3.1 could be any discrepancy between empirical measures on mini-batches e.g. maximum mean discrepancy (MMD), etc. In this case, we can see the outer optimal transport between measures over mini-batches as an additional layer to incorporate the OT property into the final loss. However, it is not easy to define the notion of a transportation plan in these cases.

C. Applications and BoMb-OT Algorithms

In this section, we collect the details of applications that mini-batch optimal transport is used in practice including deep generative models, deep domain adaptation, color transfer, and approximate Bayesian computation. Moreover, we present corresponding algorithms of these applications with our new mini-batch scheme BoMb-OT.

C.1. Deep Generative Model

Task description: We first consider the applications of the m-OT and the BoMb-OT into parametric generative modeling. The goal of parametric generative modeling is to estimate a parameter of interest, says θ, which belongs to a parameter space Θ. Each value of θ induces a model distribution μ_θ over the data space, and we want to find the optimal parameter θ^* which has μ_{θ^*} as the closest distribution to the empirical data distribution ν_n under a discrepancy (e.g. Wasserstein distance). In deep learning setting, θ is the weight of a deep neural network that maps from a low dimensional manifold Z to the data space X, and the model distribution μ_θ is a push-forward distribution $G_\theta \sharp p(z)$ for $p(z)$ is a white noise distribution on Z (e.g. $\mathcal{N}(0, I)$). By using mini-batch schemes, we can update this parameter by using the gradient of the BoMb-OT loss in Definition 3.1:

$$\theta \leftarrow Adam(\theta, \nabla_\theta \hat{D}_d^{k,m}(\mu_{\theta,n}, \nu_n)), \tag{7}$$

where $\mu_{\theta,n}$ is the empirical distribution of μ_θ. The mentioned optimization is used directly in learning generative models on the MNIST dataset with $d \equiv (SW_2, W_2^\epsilon)$ in our experiments. Since each iteration requires only a stochastic estimation of the gradient of θ, we can choose k and m to be small.

Algorithm 1 Mini-batch Deep Generative Model with BoMb-OT

Input: k, m, data $\mathcal{X}$ of size n, prior distribution $p(z)$, chosen mini-batch loss $L_{\text{DGM}} \in \{W_2, W_2^\epsilon, SW_2\}$
Initialize G_θ on GPU;
while θ does not converge **do**
  —— *On computer memory*——
  Sample indices uniformly $I_{X_1^m}, \ldots, I_{X_k^m}$ on $[n]^m$
  Sample $Z_1^m, \ldots, Z_k^m$ from $p(z)^{\otimes m}$
  Intialize $C \in \mathbb{R}^{k \times k}$
  for $i = 1$ **to** k **do**
    for $j = 1$ **to** k **do**
      —— *On GPU, autograd off*——
      Load X_i^m to GPU from $I_{X_i^m}$
      Load Z_j^m to GPU
      Compute $Y_j^m \leftarrow G_\theta(Z_j^m)$
      Compute $C_{ij} \leftarrow L_{\text{DGM}}(P_{X_i^m}, P_{Y_j^m})$
    end for
  end for
  —— *On computer memory*——
  Solve $\pi \leftarrow \text{OT}(\boldsymbol{u}_k, \boldsymbol{u}_k, C)$ (linear programing, Sinkhorn solver, etc)
  $\text{grad}_\theta \leftarrow 0$
  for $i = 1$ **to** k **do**
    for $j = 1$ **to** k **do**
      if $\pi_{ij} \neq 0$ **then**
        —— *On GPU, autograd on*——
        Load X_i^m to GPU from $I_{X_i^m}$
        Load Z_j^m to GPU
        Compute $Y_j^m \leftarrow G_\theta(Z_j^m)$
        Compute $C_{ij} \leftarrow L_{\text{DGM}}(P_{X_i^m}, P_{Y_j^m})$
        $\text{grad}_\theta \leftarrow \text{grad}_\theta + \pi_{ij} \nabla_\theta C_{ij}$
      end if
    end for
  end for
  —— *On GPU*——
  $\theta \leftarrow \text{Adam}(\theta, \text{grad}_\theta)$
end while

Algorithms: The algorithm for the deep generative model with BoMb-OT is given in Algorithm 1. This algorithm is used to train directly the generative model on the MNIST dataset.

Metric learning: Since L_2 distance is not a natural distance on the space of images such as CelebA, metric learning was introduced by (Genevay et al., 2018; Deshpande et al., 2018; Nguyen & Ho, 2022a;b) as a key step in the application of generative models. The general idea is to learn a parametric ground metric cost c_ϕ:

$$c_\phi(x, y) = \|f_\phi(x) - f_\phi(y)\|_2, \tag{8}$$

where $f_\phi : \mathcal{X} \to \mathbb{R}^h$ is a non-linear function that maps from the data space to a feature space where L_2 distance is meaningful.

The methodology to learn the function f_ϕ depends on the choice of d. For example, when $d = W_2^\epsilon$, authors in (Genevay et al., 2018) seek for ϕ by iterative updating:

$$\phi = Adam(\phi, -\nabla_\phi \hat{D}_d^{k,m}(f_\phi \sharp \mu_{\theta,n}, f_\phi \sharp \nu_n)), \tag{9}$$

In (Deshpande et al., 2018), GAN loss is used for the metric learning technique is used for $d = SW_2$.

Algorithm 2 Mini-batch Deep Domain Adaptation with BoMb-OT

Input: k, m, source domain (S, Y) of size n, target domain T of size n', chosen cost L_{DA} in Equation 11.
Initialize G_θ (parametrized by θ), F_ϕ (parametrized by ϕ)
while (θ, ϕ) do not converge **do**
  —— *On computer memory*——
  Sample indices uniformly $I_{S_1^m, Y_1^m}, \ldots, I_{S_k^m, Y_k^m}$ on $[n]^m$
  Sample indices uniformly $I_{T_1^m}, \ldots, I_{T_k^m}$ on $[n']^m$
  Initialize $C \in \mathbb{R}^{k \times k}$
  for $i = 1$ **to** k **do**
    for $j = 1$ **to** k **do**
      —— *On GPU, autograd off*——
      Load (S_i^m, Y_i^m) to GPU from $I_{S_i^m, Y_i^m}$
      Load T_j^m to GPU from $I_{T_j^m}$
      Compute $mC \leftarrow L_{\text{DA}}(S_i^m, Y_i^m, T_j^m, G_\theta, F_\phi)$ (Equation 11)
      $\boldsymbol{u}_m \leftarrow \left(\frac{1}{m}, \ldots, \frac{1}{m}\right)$
      Compute $C_{ij} \leftarrow \text{OT}(\boldsymbol{u}_m, \boldsymbol{u}_m, mC)$
    end for
  end for
  —— *On computer memory*——
  $\boldsymbol{u}_k \leftarrow \left(\frac{1}{k}, \ldots, \frac{1}{k}\right)$
  Solve $\gamma \leftarrow \text{OT}(\boldsymbol{u}_k, \boldsymbol{u}_k, C)$ (linear programing, Sinkhorn solver, etc)
  $\text{grad}_\theta \leftarrow 0$
  $\text{grad}_\phi \leftarrow 0$
  for $i = 1$ **to** k **do**
    —— *On GPU, autograd on*——
    Load (S_i^m, Y_i^m) to GPU from $I_{S_i^m, Y_i^m}$
    $\text{grad}_\theta \leftarrow \text{grad}_\theta + \frac{1}{k}\frac{1}{m}\nabla_\theta L_s(Y_i^m, F_\phi(G_\theta(S_i^m))$ (Equation 10)
    $\text{grad}_\phi \leftarrow \text{grad}_\phi + \frac{1}{k}\frac{1}{m}\nabla_\phi L_s(Y_i^m, F_\phi(G_\theta(S_i^m))$ (Equation 10)
    for $j = 1$ **to** k **do**
      if $\gamma_{ij} \neq 0$ **then**
        —— *On GPU, autograd on*——
        Load T_j^m to GPU from $I_{T_j^m}$
        Compute $mC \leftarrow C^{G_\theta, F_\phi}_{S_i^m, Y_i^m, T_j^m}$ (Equation 11)
        Compute $C_{ij} \leftarrow \text{OT}(\boldsymbol{u}_m, \boldsymbol{u}_m, mC)$
        $\text{grad}_\theta \leftarrow \text{grad}_\theta + \gamma_{ij}\nabla_\theta C_{ij}$
        $\text{grad}_\phi \leftarrow \text{grad}_\phi + \gamma_{ij}\nabla_\phi C_{ij}$
      end if
    end for
  end for
  —— *On GPU*——
  $\theta \leftarrow \text{Adam}(\theta, \text{grad}_\theta)$
  $\phi \leftarrow \text{Adam}(\theta, \text{grad}_\phi)$
end while

Learning the metric or f_ϕ is carried out simultaneously with learning the generative model in practice, namely, after one gradient step for the parameter θ, ϕ will be updated by one gradient step.

C.2. Deep Domain Adaptation

We adapt the BoMb-OT into DeepJDOT (Bhushan Damodaran et al., 2018) which is a famous unsupervised domain adaptation method based on the m-OT. In particular, we aim to learn an embedding function $G_\theta : \mathcal{X} \to \mathcal{Z}$ which maps data to the latent space; and a classifier $F_\phi : \mathcal{Z} \to \mathcal{Y}$ which maps the latent space to the label space on the target domain. For a

Algorithm 3 Color Transfer with BoMb-OT

Input: k, m, T source image $X_s \in \mathbb{R}^{n\times 3}$, target image $X_t \in \mathbb{R}^{n\times 3}$
Initialize $Y_s \in \mathbb{R}^{n\times 3}$
for $t = 1$ **to** T **do**
  Initialize $C \in \mathbb{R}^{k\times k}$
  Sample indices $I_{X_1^m}, \ldots, I_{X_k^m}$ from $[n]^m$
  Sample indices $I_{Y_1^m}, \ldots, I_{Y_k^m}$ from $[n]^m$
  for $i = 1$ **to** k **do**
    for $j = 1$ **to** k **do**
      Load X_i^m from $I_{X_i^m}$
      Load Y_i^m from $I_{Y_i^m}$
      Compute cost matrix M between X_i^m and Y_i^m
      $\pi \leftarrow \arg\min_{\pi\in\Pi(\boldsymbol{u}_m,\boldsymbol{u}_m)} \langle M, \pi\rangle$
      $C_{ij} \leftarrow \langle M, \pi\rangle$
    end for
  end for
  $\gamma \leftarrow \arg\min_{\gamma\in\Pi(\boldsymbol{u}_k,\boldsymbol{u}_k)} \langle C, \gamma\rangle$
  for $i = 1$ **to** k **do**
    for $j = 1$ **to** k **do**
      if $\gamma_{ij} \neq 0$ **then**
        Load X_i^m from $I_{X_i^m}$
        Load Y_i^m from $I_{Y_i^m}$
        Compute cost matrix M between X_i^m and Y_i^m
        $\pi \leftarrow \arg\min_{\pi\in\Pi(\boldsymbol{u}_m,\boldsymbol{u}_m)} \langle M, \pi\rangle$
        $Y_s|_{I_{X_i^m}} \leftarrow m\gamma_{ij}\ \pi \cdot X_t|_{I_{Y_j^m}}$
      end if
    end for
  end for
end for
Output: Y_s

given number of the mini-batches k and the size of mini-batches m, the goal is to minimize the following objective function:

$$\min_{G_\theta, F_\phi} L_{\text{DA}} = \Big[\frac{1}{k}\frac{1}{m}\sum_{i=1}^{k}\sum_{j=1}^{m} L_s(y_{ij}, F_\phi(G_\theta(s_{ij}))) + \min_{\gamma\in\Pi(\boldsymbol{u}_k,\boldsymbol{u}_k)} \sum_{i=1}^{k}\sum_{j=1}^{k}\gamma_{ij} \times \min_{\pi\in\Pi(\boldsymbol{u}_m,\boldsymbol{u}_m)} \langle C^{G_\theta,F_\phi}_{S_i^m,Y_i^m,T_j^m}, \pi\rangle\Big], \tag{10}$$

where $\boldsymbol{u}_k$ is the uniform measure of k supports, L_s is the source loss function (e.g. classification loss), $S_1^m, \ldots, S_k^m$ are source mini-batches which are sampled with or without replacement from the source domain $S^m \in \mathcal{X}^m$, $Y_1^m, \ldots, Y_k^m$ are corresponding labels of $S_1^m, \ldots, S_k^m$, with $S_i^m = \{s_{i1}, \ldots, s_{im}\}$ and $Y_i^m := \{y_{i1}, \ldots, y_{im}\}$. Similarly, $T_1^m, \ldots, T_k^m$ ($T_j^m := \{t_{j1}, \ldots, t_{jm}\}$) are target mini-batches which are sampled with or without replacement from the target domain $\mathcal{T}^m \in \mathcal{X}^m$. The cost matrix $C^{G,F}_{S_i^m,Y_i^m,T_j^m}$ denoted mC is defined as follows:

$$mC_{1\leq z_1,z_2\leq m} = \alpha||G(s_{iz_1}) - G(t_{jz_2})||^2 + \lambda_t L_t(y_{iz_1}, F(G(t_{jz_2}))), \tag{11}$$

where L_t is the target loss function, α and λ_t are hyper-parameters that control two terms.

Application of BoMb-UOT: For using BoMb-UOT, we only need to modify Equation 10 to:

$$\min_{G_\theta, F_\phi} L_{\text{DA}} = \Big[\frac{1}{k}\frac{1}{m}\sum_{i=1}^{k}\sum_{j=1}^{m} L_s(y_{ij}, F_\phi(G_\theta(s_{ij}))) + \min_{\gamma\in\Pi(\boldsymbol{u}_k,\boldsymbol{u}_k)} \sum_{i=1}^{k}\sum_{j=1}^{k}\gamma_{ij} \times \min_{\pi\in\mathbb{R}_+^{m\times m}} [\langle C^{G_\theta,F_\phi}_{S_i^m,Y_i^m,T_j^m}, \pi\rangle + \tau \mathrm{D}_\phi(\pi_1, \boldsymbol{u}_m) + \tau \mathrm{D}_\phi(\pi_2, \boldsymbol{u}_m)]\Big], \tag{12}$$

Algorithm 4 Approximate Bayesian Computation

Input: Generative model $p(x, \theta)$, observation $\{x_i\}_{i=1}^n$, number of iterations T, discrepancy measure D, tolerance threshold ε, number of particles m, and summary statistics s (optional).
for $t = 1$ **to** T **do**
  repeat
    Sample $\theta \sim p(\theta)$
    Sample $\{y_i\}_{i=1}^m \sim p(x|\theta)$
  until $D(s(\{y_i\}_{i=1}^m), s(\{x_i\}_{i=1}^n)) \leq \varepsilon$
  $\theta_t = \theta$
end for
Output: $\{\theta_t\}_{t=1}^T$

C.3. Color Transfer

In this appendix, we carry out more experiments on color transfer with various settings. In detail, we compare the m-OT and the BoMb-OT in different domains of images.

Methodology: In our experiment, we first compress both the source image and the target image using K-means clustering with 3000 components for the purpose of comparing with full-OT. We would like to recall that the mini-batch approach can process images with a few million pixels while the full-OT can not process those images (Fatras et al., 2020). For the m-OT and the BoMb-OT, we use the algorithm of iterative color transfer in (Fatras et al., 2020), namely, each iteration only transfers a part of the source images based on the barycentric mapping of the mini-batch transportation plan. We present the algorithm that we use in color transfer with the BoMb-OT in Algorithm 3. This algorithm is adapted from the algorithm that is used for the m-OT in (Fatras et al., 2020).

C.4. Approximate Bayesian Computation (ABC)

In this appendix, details of Approximate Bayesian Computation (ABC) and the usage of the BoMb-OT are discussed.

Review on ABC: The central task of Bayesian inference is to estimate a distribution of parameter $\theta \in \Theta$ given n data points $\{x_i\}_{i=1}^n$ and a generative model $p(x, \theta)$. Using Bayes' rule, the posterior can be written as $p(\theta|x_{1:n}) := \frac{p(x_{1:n}|\theta)p(\theta)}{p(x_{1:n})}$. Generally, the posterior is intractable since we cannot evaluate the normalizing constant $p(x_{1:n})$ (or the evidence). It leads to the usage of approximate Bayesian Inference, e.g., Markov Chain Monte Carlo and Variational Inference. However, in some settings, the likelihood function $p(x_{1:n}|\theta)$ cannot be evaluated such as implicit generative models. In these cases, Approximate Bayesian Computation (or likelihood-free inference) is a good framework to infer the posterior distribution since it only requires the samples from the likelihood function.

We present the algorithm of ABC in Algorithm 4 that is used to obtain posterior samples of θ. The thing that sets ABC apart is that it can be implemented in distributed ways, and its posterior can be shown to have the desirable theoretical property of converging to the true posterior when $\varepsilon \to 0$. However, the performance of ABC depends on the choice of the summary statistics s (e.g., empirical mean and empirical variance) and the discrepancy D. In practice, constructing sufficient statistics is not easy. Thus, a discrepancy between empirical distributions is used to avoid this non-trivial task. Currently, Wasserstein distances have drawn a lot of attention from ABC's community because they provide a meaningful comparison between non-overlap probability distributions (Bernton et al., 2019a; Nadjahi et al., 2020). When the number of particles m in Algorithm 4 is the largest OT problem that can be solved by the current computational resources, the mini-batch approach can be utilized to obtain more information about two measures, namely, more supports in sample space can be used. In particular, we utilize the mini-batch OT losses as the discrepancy D in Algorithm 4. The detail of the application of BoMb-OT in ABC is given in Algorithm 5.

D. Additional Experiments

In this appendix, we provide additional experimental results that are not shown in the main paper. In Appendix D.1, we present randomly generated images on the MNIST dataset, CIFAR10 dataset, CelebA dataset, and training time comparison between the m-OT and the BoMb-OT. Furthermore, we give detailed results on deep domain adaptation in Appendix D.2

Algorithm 5 Approximate Bayesian Computation with BoMb-OT

Input: Generative model $p(x, \theta)$, observation $\{x_i\}_{i=1}^n$, number of iterations T, optimal transport discrepancy measure D_{OT}, tolerance threshold ε, number of particles m, number of mini-batches k, and summary statistics s (optional).
for $t = 1$ **to** T **do**
  repeat
    Sample $\theta \sim p(\theta)$
    Sample indices $I_{X_1^m}, \ldots, I_{X_k^m}$ from $[n]^m$
    Sample $\{y_i\}_{i=1}^n \sim p(x|\theta)$
    Sample indices $I_{Y_1^m}, \ldots, I_{Y_k^m}$ from $[n]^m$
    Initialize $C \in \mathbb{R}^{k \times k}$
    for $i = 1$ **to** k **do**
      for $j = 1$ **to** k **do**
        Load X_i^m, Y_j^m from their indices $I_{X_i^m}, I_{Y_j^m}$
        $C_{ij} \leftarrow D_{\text{OT}}(X_i^m, Y_j^m)$
      end for
    end for
    $\boldsymbol{u}_k \leftarrow \left(\frac{1}{k}, \ldots, \frac{1}{k}\right)$
    $D \leftarrow \text{OT}(\boldsymbol{u}_k, \boldsymbol{u}_k, C)$
  until $D \leq \varepsilon$
  $\theta_t = \theta$
end for
Output: $\{\theta_t\}_{t=1}^T$

and we also provide the computational time. In Appendix D.3 we carry out more experiments on color transformation with various settings on different images and their corresponding color palettes. After that, we compare the m-OT and the BoMb-OT on gradient flow application in Appendix D.4. Finally, we investigate the behavior of the m-OT and the (e)BoMb-OT in estimating the optimal transportation plan in Appendix D.5.

D.1. Deep Generative model

Generated images: We show the generated images on the MNIST dataset in Figure 4, the generated images on CelebA in Figure 5, and the generated images on CIFAR10 in Figure 6. For both choices of d (SW_2, W_2^ϵ), we can see that the images from BoMb-OT are more realistic than the images from m-OT. This qualitative result supports the quantitative in Table 1 in the main text.

Computational speed: Table 5 details the computational speed of the deep generative model when using mini-batch size $m = 100$. In general, the more the number of mini-batches is, the more complex the problem is. Increasing the number of mini-batches k does decrease the number of iterations per second in all experiments. For both datasets, using the sliced Wasserstein distance ($d = SW_2$) leads to higher iterations per second compared to the entropic Wasserstein distance ($d = W_2^\epsilon$) on the high dimensional space. This result is expected since the slicing approach aims to reduce computational complexity. On the MNIST dataset, we observe that the entropic regularized version of the BoMb-OT ($\lambda > 0$) is the slowest for both choices of d. The m-OT is the fastest approach with the number of iterations per second of $7.89, 2.54$, and 0.85 for $k = 2, 4$, and 8 respectively. In contrast, the BoMb-OT ($\lambda = 0$) is faster than the m-OT for CIFAR10 and CelebA datasets because we set a high value for the regularized parameter (e.g. $\epsilon = 50, 40$). For the CelebA dataset, although the BoMb-OT ($\lambda > 0$) is the slowest method when $d = SW_2$, it becomes the fastest approach if we set d to W_2^ϵ. The reason for this phenomenon is because of the sparsity of the BoMb-OT's transportation plan between measures over mini-batches. In particular, pairs of mini-batches that are zero can be skipped (see in Algorithm 1. So, the BoMb-OT can avoid estimating the gradient of the neural networks of a bad pair of mini-batches.

D.2. Deep domain adaptation

In this section, we compare the performance of two mini-batch schemes on digits and Office-Home datasets.

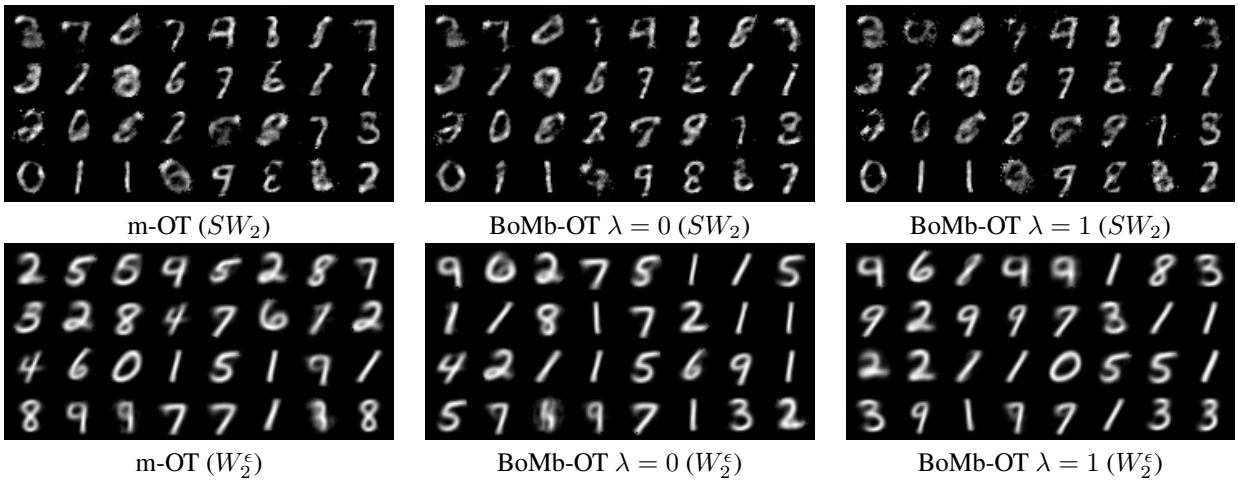

Figure 4. MNIST generated images from the m-OT and the BoMb-OT for $(k, m) = (4, 100)$.

Figure 5. CelebA generated images from the m-OT and the BoMb-OT for (k,m)=(4,100).

Ablation study on the mini-batch size: In this experiment, we study the effect of changing the mini-batch size on the target domain classification accuracy. The number of mini-batches k is fixed to 32 while the mini-batch size m varies from 10 to 50 on the SVHN to MNIST task and from 5 to 20 on USPS to MNIST task (because the USPS dataset has a much smaller number of samples than the SVHN dataset). The number of epochs also varies so that the same number of stochastic gradient updates is applied in each case. As seen in Table 6, the BoMb-OT produces better classification accuracy than the m-OT in all experiments. In addition, it leads to a huge performance improvement (over 19%) in comparison with the m-OT when the mini-batch size is small ($m = 10$ for SVHN to MNIST and $m = 5$ for USPS to MNIST). When the mini-batch size becomes larger, the performance of both methods also increases. An explanation for such an outcome is that a large mini-batch size makes the mini-batch loss approach to full-OT.

Office-Home dataset: Table 7 illustrates the performance of two mini-batch schemes on the Office-Home dataset when changing the number of mini-batches k from 1 to 4. When $k = 2, 4$, BoMb-UOT achieves the classification accuracy higher than m-UOT on 12 out of 12 scenarios, resulting in an improvement of 0.25 on average. Using 4 mini-batches, BoMb-OT also improves the performance over m-OT on all domain adaptation tasks with an average gap of 0.57. Similar to digits datasets, we observe that the classification accuracy on the target domain of m-OT and BoMb-OT tends to improve as k increases from 1 to 4 while m-UOT and BoMb-UOT do not show the same tendency.

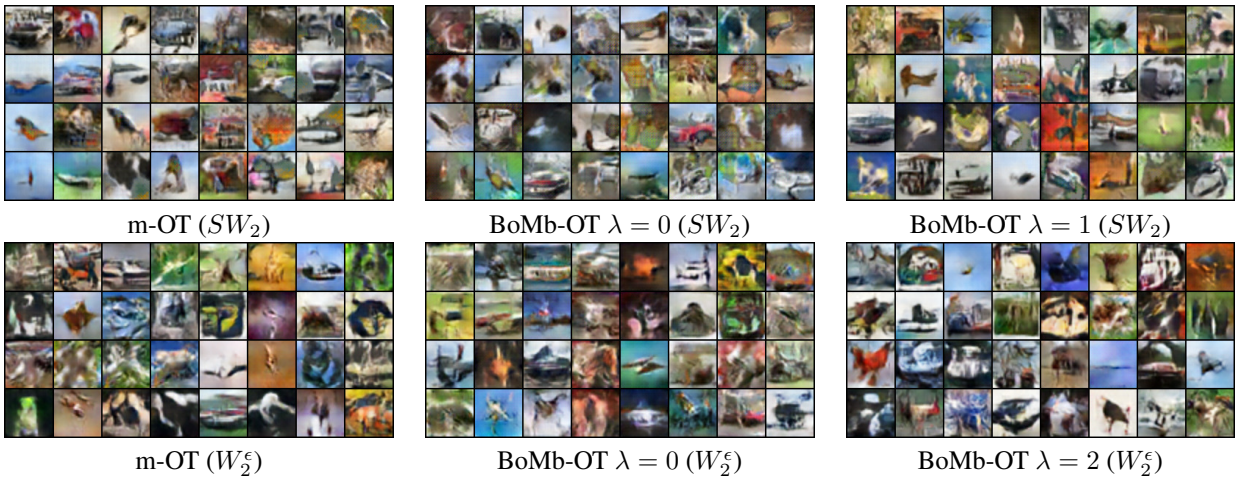

Figure 6. CIFAR10 generated images from the m-OT and the BoMb-OT for (k,m)=(4,100).

Table 5. Number of iterations per second of deep generative models.

Dataset	k	m-OT(W_2^ϵ)	BoMb-OT $\lambda = 0$ (W_2^ϵ)	BoMb-OT $\lambda > 0$ (W_2^ϵ)	m-OT(SW_2^ϵ)	BoMb-OT $\lambda = 0$ (SW_2^ϵ)	BoMb-OT $\lambda > 0$ (SW_2^ϵ)
MNIST	1	27.27	27.27	27.27	54.55	54.55	54.55
	2	**7.89**	5.88	4.62	**18.75**	15.00	11.54
	4	**2.54**	2.11	1.36	**6.00**	5.36	3.41
	8	**0.85**	0.79	0.44	**1.70**	1.74	0.95
CIFAR10	1	22.73	22.73	22.73	25.00	25.00	25.00
	2	6.94	5.81	**7.35**	9.62	**10.00**	7.58
	4	2.05	**2.31**	2.16	3.21	**4.17**	2.40
	8	0.53	**0.78**	0.59	0.97	**1.54**	0.71
CelebA	1	6.78	6.78	6.78	9.93	9.93	9.93
	2	1.92	1.75	**2.52**	3.38	**3.55**	2.65
	4	0.45	0.54	**0.68**	1.03	**1.40**	0.78
	8	0.11	0.15	**0.18**	0.29	**0.51**	0.22

Table 6. Effect of changing the mini-batch size on the performance of two mini-batch schemes on the deep domain adaptation application.

Scenario	m	k	Number of epochs	m-OT	BoMb-OT $\lambda = 0$	Improvement
SVHN to MNIST	10	32	64	69.41	**88.49**	+19.08
	20	32	128	89.69	**93.54**	+3.85
	50	32	320	93.09	**95.59**	+2.40
USPS to MNIST	5	32	50	73.92	**93.15**	+19.23
	10	32	100	92.96	**95.06**	+2.10
	20	32	200	95.85	**96.15**	+0.30

Computational speed: The number of iterations per second of deep DA can be found in Tables 8-9. Similar to the deep generative model, we observe a phenomenon that the speed of both the m-OT and BoMb-OT decreases as k increases. Interestingly, increasing the mini-batch size m when adapting from USPS to MNIST barely affects the running speed of both methods. The run time of the m-OT nearly doubles that of the BoMb-OT. Specifically, the BoMb-OT averagely runs 1 iteration in a second while an iteration of the m-OT consumes roughly 2 seconds. Although having comparable time complexity, the BoMb-OT in practice runs faster than the m-OT in all deep DA experiments. This is because of the sparsity

Table 7. Comparison between the two mini-batch schemes on deep domain adaptation on Office-Home datasets. Experiments were run 3 times. Each entry includes the average accuracy and its standard deviation over 3 runs.

k	Methods	A2C	A2P	A2R	C2A	C2P	C2R	P2A	P2C	P2R	R2A	R2C	R2P	Avg
1	m-OT	49.50 ± 0.61	66.22 ± 0.84	73.41 ± 0.76	57.38 ± 0.32	64.50 ± 0.36	67.17 ± 0.23	54.95 ± 0.44	47.07 ± 0.50	73.33 ± 0.32	64.51 ± 0.42	53.72 ± 0.58	76.88 ± 0.10	62.39
	BoMb-OT	49.50 ± 0.61	66.22 ± 0.84	73.41 ± 0.76	57.38 ± 0.32	64.50 ± 0.36	67.17 ± 0.23	54.95 ± 0.44	47.07 ± 0.50	73.33 ± 0.32	64.51 ± 0.42	53.72 ± 0.58	76.88 ± 0.10	62.39
	m-UOT	54.99 ± 0.41	74.45 ± 0.16	80.78 ± 0.34	65.66 ± 0.20	74.93 ± 0.39	74.91 ± 0.13	64.70 ± 0.87	53.42 ± 0.26	80.01 ± 0.12	74.58 ± 0.56	59.88 ± 0.30	83.73 ± 0.12	70.17
	BoMb-UOT	54.99 ± 0.41	74.45 ± 0.16	80.78 ± 0.34	65.66 ± 0.20	74.93 ± 0.39	74.91 ± 0.13	64.70 ± 0.87	53.42 ± 0.26	80.01 ± 0.12	74.58 ± 0.56	59.88 ± 0.30	83.73 ± 0.12	70.17
2	m-OT	49.76 ± 0.30	68.37 ± 0.16	74.40 ± 0.42	59.64 ± 0.34	**64.69** ± 0.20	**68.63** ± 0.18	56.12 ± 0.32	46.69 ± 0.33	**74.37** ± 0.21	67.27 ± 0.46	54.45 ± 1.28	77.95 ± 0.25	63.53
	BoMb-OT	**50.02** ± 0.06	**68.47** ± 0.17	**74.64** ± 0.29	**59.86** ± 0.47	64.66 ± 0.15	68.41 ± 0.21	**56.32** ± 0.38	**47.05** ± 0.06	74.36 ± 0.09	**67.74** ± 0.82	**54.52** ± 0.82	**78.19** ± 0.06	**63.69**
	m-UOT	56.08 ± 0.25	74.90 ± 0.16	80.47 ± 0.11	65.57 ± 0.54	73.96 ± 0.40	74.88 ± 0.14	65.87 ± 0.02	52.98 ± 0.33	79.97 ± 0.14	74.19 ± 0.15	59.87 ± 0.17	83.10 ± 0.22	70.15
	BoMb-UOT	**56.23** ± 0.24	**75.24** ± 0.09	**80.53** ± 0.04	**65.80** ± 0.21	**74.57** ± 0.30	**75.38** ± 0.19	**66.15** ± 0.12	**53.21** ± 0.16	**80.03** ± 0.05	**74.25** ± 0.12	**60.12** ± 0.08	**83.30** ± 0.06	**70.40**
4	m-OT	49.80 ± 0.16	68.82 ± 0.56	74.57 ± 0.07	59.24 ± 0.43	65.08 ± 0.04	68.15 ± 0.13	56.63 ± 0.23	46.82 ± 0.06	74.04 ± 0.25	67.22 ± 0.14	53.80 ± 0.34	78.11 ± 0.02	63.52
	BoMb-OT	**50.16** ± 0.32	**69.57** ± 0.69	**74.84** ± 0.10	**60.24** ± 0.37	**65.18** ± 0.27	**69.15** ± 0.49	**57.48** ± 0.15	**47.42** ± 0.14	**74.88** ± 0.11	**67.39** ± 0.17	**54.19** ± 0.30	**78.59** ± 0.15	**64.09**
	m-UOT	55.52 ± 0.12	75.20 ± 0.05	80.47 ± 0.03	65.84 ± 0.06	74.02 ± 0.10	74.85 ± 0.25	65.64 ± 0.09	52.83 ± 0.22	79.96 ± 0.03	74.14 ± 0.13	59.89 ± 0.08	83.07 ± 0.09	70.12
	BoMb-UOT	**56.22** ± 0.18	**75.27** ± 0.09	**80.56** ± 0.13	**66.01** ± 0.33	**74.23** ± 0.26	**75.19** ± 0.09	**66.02** ± 0.14	**53.17** ± 0.08	**80.05** ± 0.09	**74.32** ± 0.04	**60.07** ± 0.15	**83.32** ± 0.09	**70.37**

Table 8. Computational speed of deep DA when changing k.

Scenario	k	m	m-OT	BoMb-OT $\lambda = 0$
SVHN to MNIST	8	50	2.26	**3.05**
	16	50	0.60	**1.06**
	32	50	0.15	**0.32**
USPS to MNIST	8	25	6.00	**9.00**
	16	25	1.80	**2.57**
	32	25	0.45	**0.75**

Table 9. Computational speed of deep DA when changing m.

Scenario	m	k	m-OT	BoMb-OT $\lambda = 0$
SVHN to MNIST	10	32	0.33	**0.61**
	20	32	0.33	**0.60**
	50	32	0.15	**0.32**
USPS to MNIST	5	32	0.49	**1.00**
	10	32	0.49	**1.00**
	20	32	0.46	**1.00**

of the BoMb-OT's transportation plan between measures over mini-batches. In particular, pairs of mini-batches that are zero can be skipped (see in Algorithm 2). We would like to recall that gradient estimation is the most time-consuming task in deep learning applications.

D.3. Color Transfer

In this appendix, we compare the m-OT and the BoMb-OT in different domains of images including natural images and arts.

Comparison between the m-OT and the BoMb-OT: We illustrate the color-transferred images using the m-OT and the BoMb-OT with two values of the number of mini-batches and the size of mini-batches $(k, m) = (10, 10)$ and $(k, m) = (20, 20)$. The number of incremental step T (see in Algorithm 3) is set to 5000. We show the source images, target images, and corresponding transferred images in Figure 7-Figure 8. It is easy to see that the transferred images from the BoMb-OT look more realistic than the m-OT, and the color is more similar to the target images. The color palette of transferred images also reinforces the above claim when it is closer to the color palette of the target images. According to those figures, increasing the number of mini-batches and the mini-batch size improves the results considerably.

Computational speed: For $k = 10, m = 10$, the m-OT has the speed of 103 iterations per second while the BoMb-OT has the speed of 100 iterations per second. For $k = 20, m = 20$, the speed of m-OT is about 20 iterations per second and the

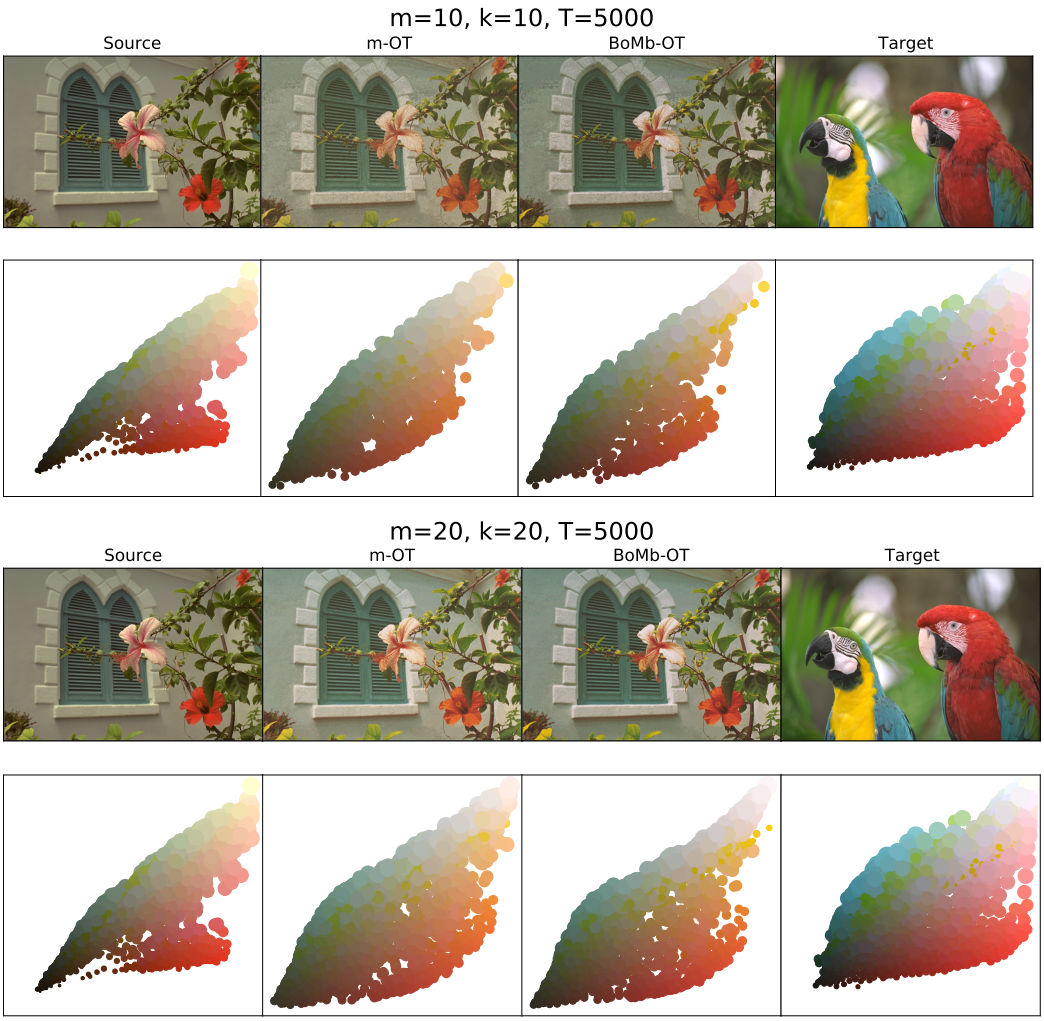

Figure 7. Experimental results on color transfer for the m-OT and the BoMb-OT on natural images with $(k, m) = (10, 10)$, $(k, m) = (20, 20)$, and $T = 5000$. Color palettes are shown under corresponding images.

speed of the BoMb-OT is also about 20 iterations per second. It means that the additional $k \times k$ OT is not too expensive while it can improve the color palette considerably.

Comparison to full-OT: In contrast to other applications where the full-OT is not applicable due to a very big number of supports n or continuous measures, we can run full-OT here since we have used K-mean to reduce n to 3000. We show the result in Figure 9. From the figure, we can see that the full-OT gives a perfect transferred color palette. However, we would like to recall that full-OT cannot be used when n is large (Fatras et al., 2020) that often happens in practice. Therefore, using BoMb-OT is a good alternative for the full-OT due to the almost similar performance.

D.4. Non-parametric generative model via gradient flow

In this appendix, we show the experiment that uses the m-OT, the BoMb-OT, and the entropic regularized BoMb-OT (eBoMb-OT) in the gradient flow application.

Task description: Gradient flow is a non-parametric method to learn a generative model. Like every generative model, the goal of gradient flow is to mimic the data distribution ν by a distribution μ. It leads to the functional optimization problem:

$$\min_{\mu} D(\mu, \nu), \tag{13}$$

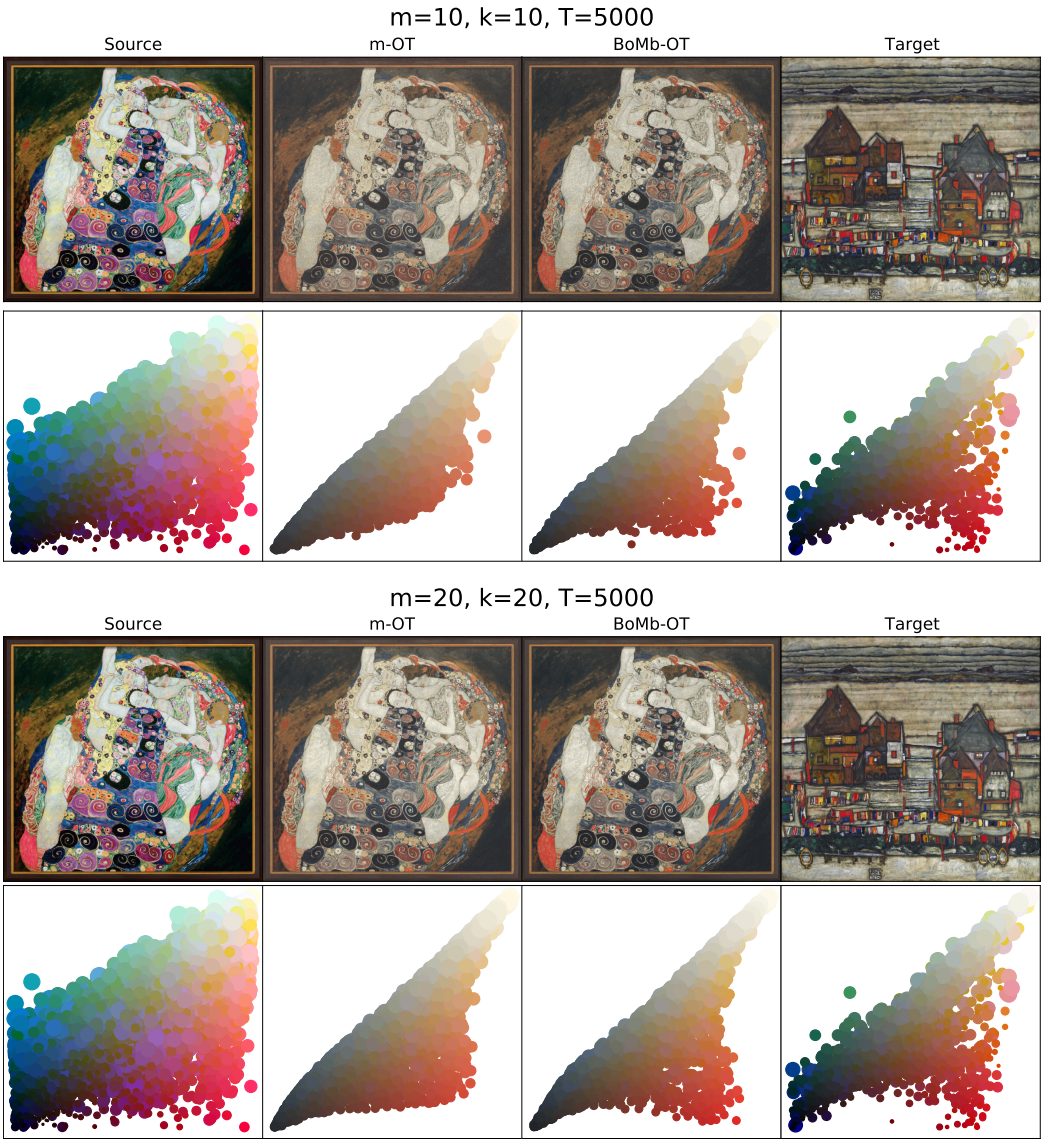

Figure 8. Experimental results on color transfer for the m-OT and the BoMb-OT on arts with $(k, m) = (10, 10)$, $(k, m) = (20, 20)$, and $T = 5000$. Color palettes are shown under corresponding images.

where D is a predefined discrepancy between two probability measures. So, a gradient flow can be constructed:

$$\partial_t \mu_t = -\nabla_{\mu_t} D(\mu_t, \nu) \tag{14}$$

We follow the Euler scheme to solve this equation as in (Feydy et al., 2019), starting from an initial distribution at time $t = 0$. In this paper, we choose D to be BoMb-OT (W_2) and the m-OT (W_2) for the sake of comparison between them.

We first consider the toy example as in (Feydy et al., 2019) and present our results in Figure 10. The task is to move the colorful empirical measure to the "S-shape" measure. Each measure has 1000 support points. Here, we choose $(k, m) = (4, 16)$, the OT loss is Wasserstein-2, and we use the Wasserstein-2 score to evaluate the performance of the mini-batch scheme. From Figure 10, BoMb-OT provides better flows than m-OT, namely, Wasserstein-2 scores of BoMb-OT are always lower than those of m-OT in every step. In addition, we do an extra setup with a higher of mini-batches, $(k, m) = (16, 16)$ to show that increasing the number of mini-batches improves the performance of both m-OT, BoMb-OT, and entropic regularized BoMb-OT. The result is shown in Figure 11. In this setting, the BoMb-OT still shows its favorable performance compared to the m-OT, namely, its Wasserstein-2 scores are still lower than the m-OT in every step.

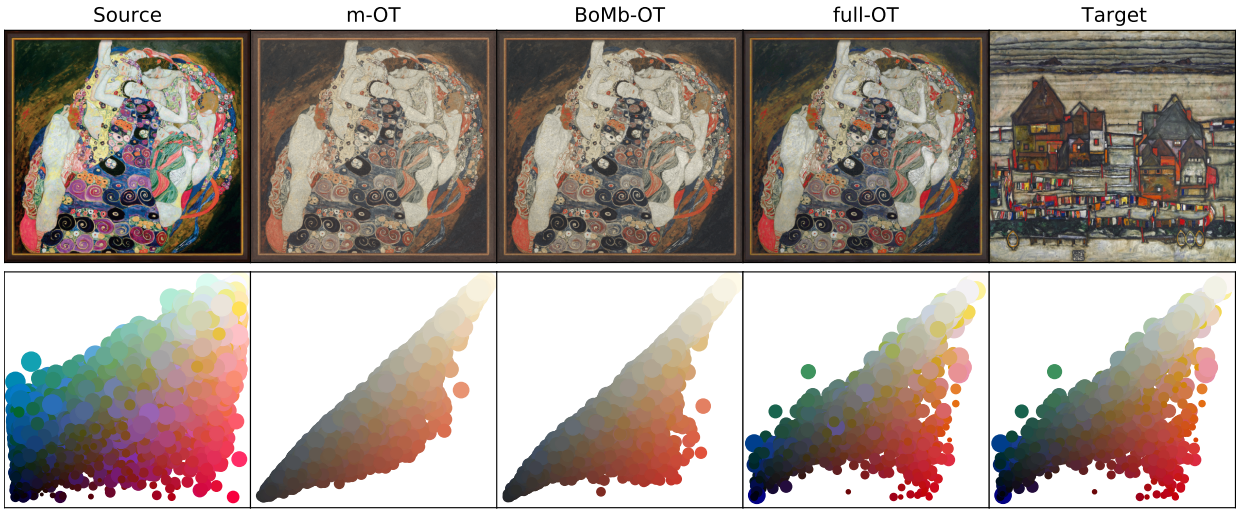

Figure 9. Experimental results on color transfer for full OT, the m-OT, and the BoMb-OT on natural images with $(k, m) = (10, 10)$, and $T = 5000$. Color palettes are shown under corresponding images.

CelebA: Let μ and ν denote the empirical measures defined over 5000 female images and 5000 male images in the CelebA dataset. Thus, we can present the transformation from a male face to be a female one by creating a flow from μ to ν. Our setting experiment is the same as (Fatras et al., 2020). We first train an autoencoder on CelebA, then we compress two original measures to the low-dimensional measures on the latent space of the autoencoder. After having the latent measures, we run the Euler scheme to get the transformed measure then we decode it back to the data space by the autoencoder. The result is shown in Figure 12, we also show the closest female image (in sense of L_2 distance) to the final found image in each method and the corresponding L_2 distance between the middle step images and the nearest image. As shown, (e)BoMb-OT provides a better flow than m-OT does.

D.5. Transportation Plan and Transportation Cost

In this appendix, we investigate deeper the behavior of the m-OT and the BoMb-OT in estimating the transportation plan. We present a toy example to illustrate the transportation plans of the m-OT, the BoMb-OT, and the original OT (full-OT).

Transportation cost and Computational Speed: We consider two empirical of 1000 samples from $\mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$ and $\mathcal{N}\left(\begin{bmatrix}4\\4\end{bmatrix}, \begin{bmatrix}1 & -0.8\\-0.8 & 1\end{bmatrix}\right)$. We compute the value and the computational speed of the m-OT, the BoMb-OT with $m \in \{10, 100\}$ and $k \in \{1, 50, 100, 150, 200, 250, 300\}$, then plot them with the value of the speed of full-OT in Figure 13. From the figure, we can see that the cost of the BoMb-OT is always lower than m-OT and is closer to the full-OT. For the computational speed, the BoMb-OT is comparable to m-OT. We would like to recall that the main reason that full-OT cannot be used in practice is because of the memory issue.

Transportation plans: We sample two empirical distributions μ_n and ν_n, where $n = 10$, from $\mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$ and $\mathcal{N}\left(\begin{bmatrix}4\\4\end{bmatrix}, \begin{bmatrix}1 & -0.8\\-0.8 & 1\end{bmatrix}\right)$ respectively. We plot the graph of sample matchings and transportation matrices in Figure 14. When the size of mini-batches equals 2 and the number of mini-batches is set to 20, the m-OT approach produces messy OT matrices and disordered matching, meanwhile, the BoMb-OT can still concentrate the mass to meaningful entries of the transportation matrix, thus, its matching is acceptable. Increasing the size of mini-batches to 8, m-OT's performance improves significantly however its matching and its matrices are visibly still not good solutions. In contrast, the BoMb-OT is able to generate nearly optimal matchings and transportation matrices. Next, we test the m-OT and the BoMb-OT in real applications in the case of an extremely small number of mini-batches. When there are 8 mini-batches of size 2, the m-OT still performs poorly while the BoMb-OT creates a sparser transportation matrix that is closer to the optimal solution obtained by the full-OT. Similarly, with 2 mini-batches of size 8, the BoMb-OT has only 4 wrong matchings while that number of the m-OT is 10. In conclusion, the BoMb-OT is the better version for mini-batch with any value of the number of mini-batches and mini-batches size.

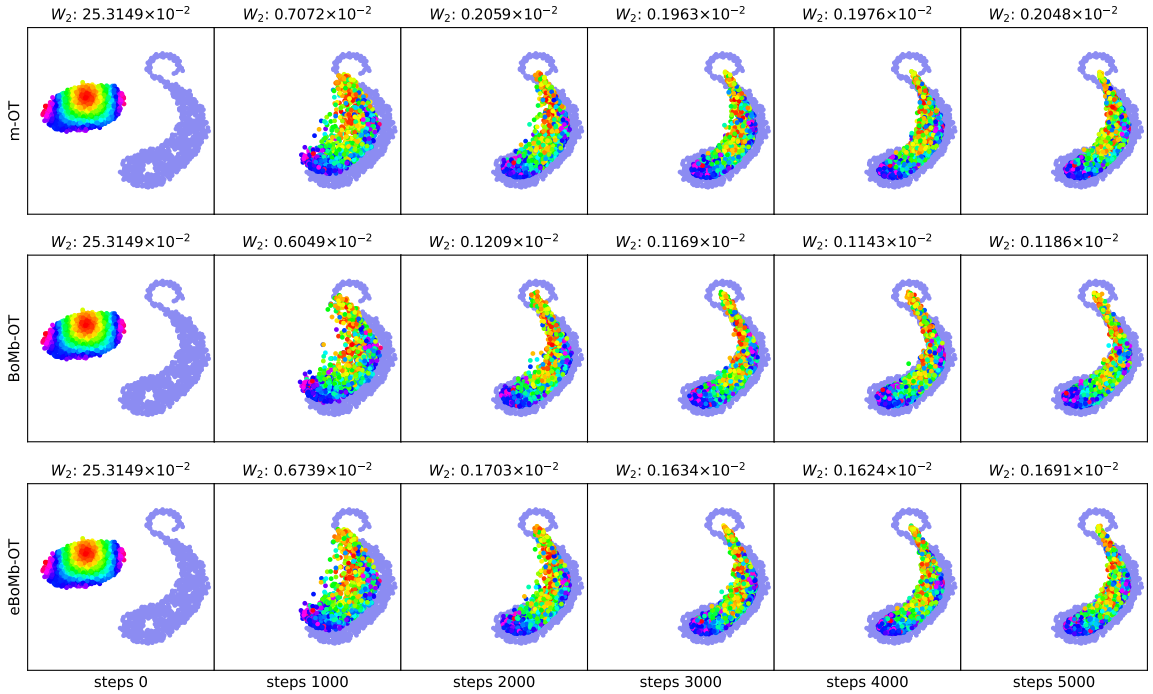

Figure 10. Comparison between the BoMb-OT and the m-OT in gradient flow with $n = 1000$, $k = 4$, $m = 16$. In the entropic regularized parameter of BoMb-OT (eBoMb-OT), λ is set to 0.01.

Entropic transportation plan of the BoMb-OT: We empirically show that the transportation plan of the BoMb-OT, when the entropic regularization is sufficiently large, reverts into the m-OT's plan. The result is given in Figure 15. When $\lambda = 10^3$, the transportation plans of the BoMb-OT are identical to plans of the m-OT with every choice of (k, m). However, when $\lambda = 0.1$, despite not being identical to full-OT, the plans produced by the BoMb-OT are still close to the true optimal plan.

Comparison with stochastic averaged gradient (SAG): We compare the BoMb-OT with the stochastic averaged gradient (SAG) (Aude et al., 2016) for computing OT. The transportation matrices are given in Figure 16. We would like to recall that SAG still needs to store the full cost matrix ($n\times$) while the BoMb-OT only needs to store smaller cost matrices ($m \times m$). We can see that the BoMb-OT gives a better transportation plan than SAG when $m = 2$. When $m = 8$, the BoMb-OT still seems to be better.

E. Experiment Settings

In this appendix, we collect some necessary experimental setups in the paper including generative model, gradient flow. We use POT (Flamary et al., 2021) for OT solvers and the pyABC (Klinger et al., 2018) for the ABC experiments. In our experiments, we create mini-batches by sampling without replacement from supports of the original empirical measures. For continuous measures, we sample i.i.d random variables to form mini-batches.

E.1. Deep Generative model

Wasserstein-2 scores: We use empirical distribution with 11000 samples that are obtained by sampling from the generative model and the empirical test set distribution respectively, then we compute discrete Wasserstein-2 distance via linear programming.

FID scores: We use 11000 samples from the generative model and all test set images to compute the FID score (Heusel et al., 2017).

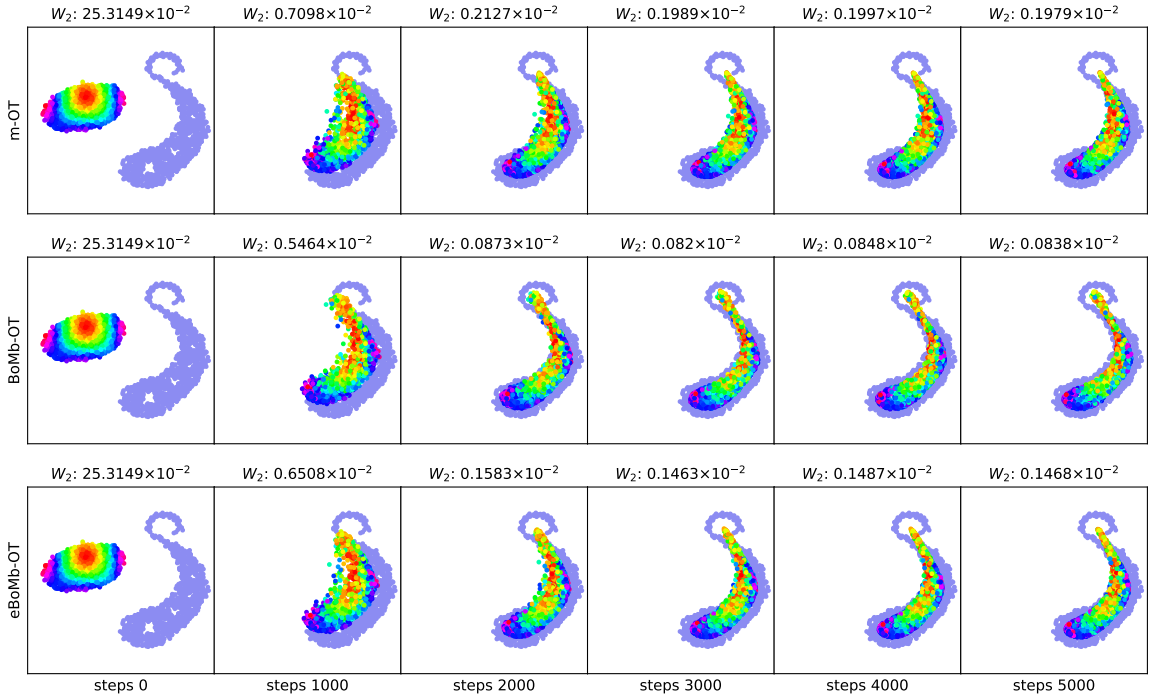

Figure 11. Visualization of the gradient flow provided by the m-OT and the BoMb-OT with corresponding Wasserstein-2 scores ($n = 1000, k = 16, m = 16$). In the entropic regularized parameter of BoMb-OT (eBoMb-OT), λ is set to 0.01.

Parameter settings: We chose a learning rate equal to 0.0005, batch size equal to 100, number of epochs in MNIST equal to 100, number of epochs in CelebA equal to 25, number of epochs in CIFAR10 equal to 50. For $d = SW_2$ we use the number of projections $L = 1000$ on MNIST and $L = 100$ on CIFAR10 and CelebA. For $d = W_2^\epsilon$, we use $\epsilon = 1$ for MNIST, $\epsilon = 50$ for CIFAR10, $\epsilon = 40$ for CelebA. For the entropic regularized version of BoMb-OT, we choose the best setting for $\lambda \in \{1, 2, 3, 4, 5, 10, 20, 30, 40, 50, 60, 70, 80\}$

Neural network architectures: We use the MLP for the generative model on the MNIST dataset, while CNNs are used on the CelebA dataset.

Generator architecture was used for MNIST:
$z \in \mathbb{R}^{32} \to FC_{100} \to ReLU \to FC_{200} \to ReLU \to FC_{400} \to ReLU \to FC_{784} \to ReLU$

Generator architecture was used for CelebA: $z \in \mathbb{R}^{128} \to TransposeConv_{512} \to BatchNorm \to ReLU \to TransposeConv_{256} \to BatchNorm \to ReLU \to TransposeConv_{128} \to BatchNorm \to ReLU \to TransposeConv_{64} \to BatchNorm \to ReLU \to TransposeConv_{1} \to Tanh$
Metric learning neural network's architecture was used for CelebA:

First part: $x \in \mathbb{R}^{64\times64\times3} \to Conv_{64} \to LeakyReLU_{0.2} \to Conv_{128} \to BatchNorm \to LearkyReLU_{0.2} \to Conv_{256} \to BatchNorm \to LearkyReLU_{0.2} \to Conv_{512} \to BatchNorm \to Tanh$
Second part: $Conv_1 \to Sigmoid$
Generator architecture was used for CIFAR10: $z \in \mathbb{R}^{128} \to TransposeConv_{256} \to BatchNorm \to ReLU \to TransposeConv_{128} \to BatchNorm \to ReLU \to TransposeConv_{64} \to BatchNorm \to ReLU \to TransposeConv_{1} \to Tanh$
Metric learning neural network's architecture was used for CIFAR:

First part: $x \in \mathbb{R}^{32\times32\times3} \to Conv_{64} \to LeakyReLU_{0.2} \to Conv_{128} \to BatchNorm \to LearkyReLU_{0.2} \to Conv_{256} \to BatchNorm \to Tanh$
Second part: $Conv_1 \to Sigmoid$

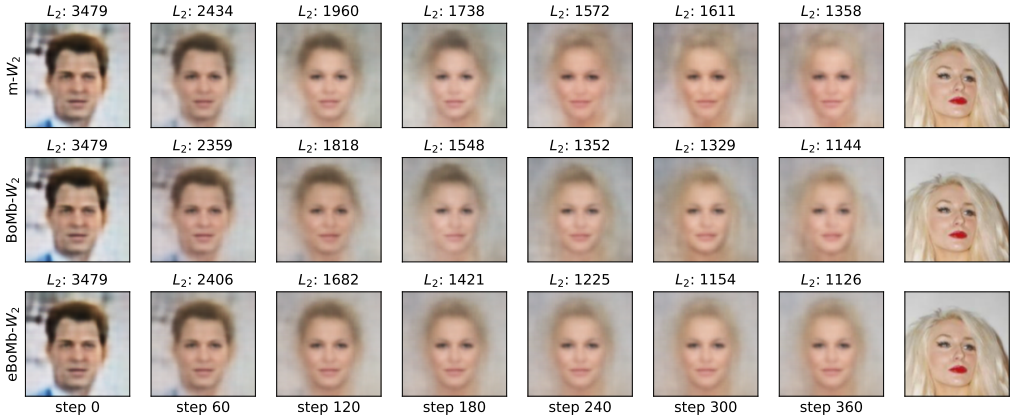

Figure 12. Transforming man face to woman face by using the gradient flow with the m-OT, the BoMb-OT, the entropic BoMb-OT (eBoMb-OT) on CelebA dataset for $(k, m) = (25, 100)$. The last column is the nearest image (in sense of L_2 distance) to the final found female image, the corresponding L_2 distances are also shown on the top of middle-stage images.

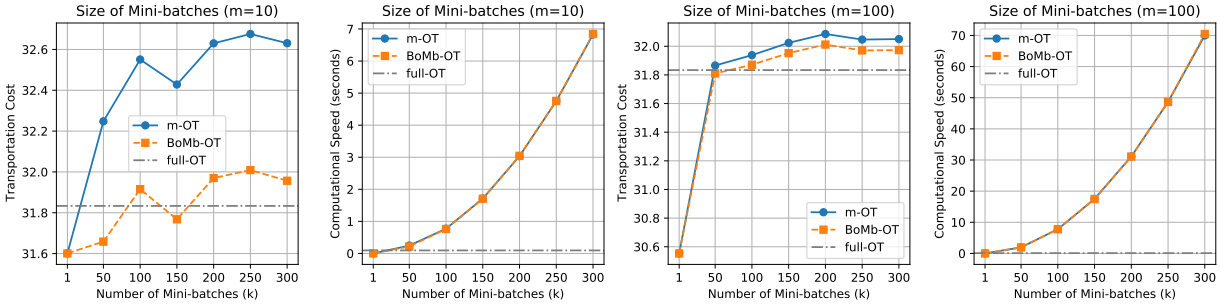

Figure 13. Comparison between the m-OT and the BoMb-OT in terms of computational speed and value with different values of k. The red horizontal line is the full-OT between original measures.

E.2. Deep domain adaptation

In this section, we state the neural network architectures and hyper-parameters for deep domain adaptation.

Evaluation metric: The classification accuracy is utilized to evaluate the mini-batch methods.

Parameter settings for digits datasets: The number of mini-batches k varies in $\{1, 2, 4, 8\}$. For the SVHN dataset, the mini-batch size m is set to 50. Because the USPS dataset has fewer samples than the SVHN dataset, m is set to 25. We train the network using Adam optimizer with an initial learning rate of 0.0002. The number of epochs is 80 for $k = 8$. As the number of mini-batches doubles, we double the number of epochs so that the number of iterations does not change. The hyperparameters of m-OT and BoMb-OT in Equation 11 follow the settings in DeepJDOT: $\alpha = 0.001, \lambda_t = 0.0001$. The hyperparameters of m-UOT and BoMb-UOT are the same as JUMBOT: $\alpha = 0.1, \lambda_t = 0.1, \epsilon = 0.1, \tau = 1$. For computing BoMb-OT and BoMb-UOT, we simply set the value of λ to 0.

Parameter settings for the VisDA dataset: All methods are trained using a batch size of 72 during 10000 iterations. The number of mini-batches k for each method is selected from the set $\{1, 2, 4\}$. Following the settings in (Fatras et al., 2021a), the coefficients in the cost formula is as follows $\alpha = 0.005, \lambda_t = 1, \tau = 0.3, \epsilon = 0.01$. For computing BoMb-OT and BoMb-UOT, we simply set the value of λ to 0.

Parameter settings for the Office-Home dataset: The number of mini-batches k varies in $\{1, 2, 4\}$. Following the settings in (Fatras et al., 2021a), we train models with a mini-batch size $m = 65$ during 10000 iterations. The hyperparameters for computing the cost matrix follow the settings in JUMBOT: $\alpha = 0.01, \lambda_t = 0.5, \tau = 0.5, \epsilon = 0.01$. For computing BoMb-OT and BoMb-UOT, we choose the best value of $\lambda \in \{0, 0.01, 0.1, 1, 10, 100, 1000\}$.

Training details: Similar to both DeepJDOT and JUMBOT, we stratify the data loaders so that each class has the same

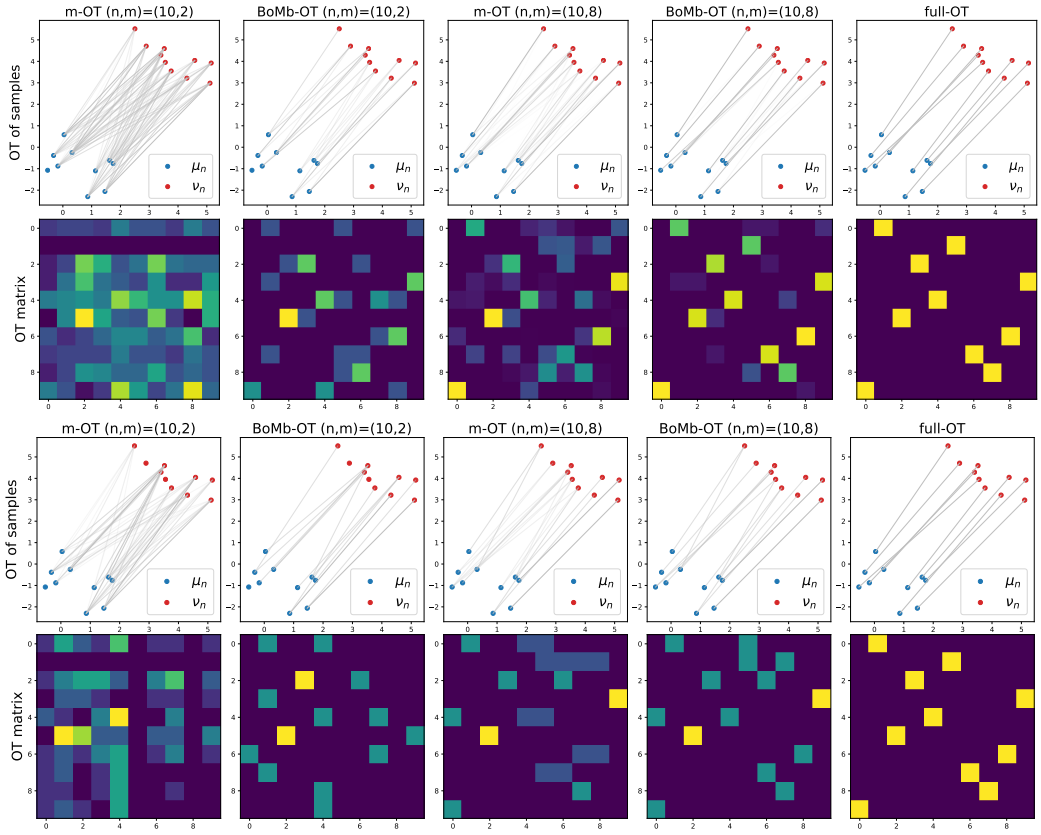

Figure 14. Visualization of transportation plans of the m-OT and the BoMb-OT between 2D empirical distributions with 10 samples. The first two rows are results of $k = 20$. The next two rows are for $(k, m) = (8, 2)$ and $(k, m) = (2, 8)$ in turn.

number of samples in the mini-batches. For digits datasets, we also train our neural network on the source domain during 10 epochs before applying our method. For VisDA and Office-home, because the classifiers are trained from scratch, their learning rates are set to be 10 times that of the generator. We optimize the models using an SGD optimizer with momentum = 0.9 and weight decay = 0.0005. We schedule the learning rate with the same strategy used in JUMBOT. The learning rate at iteration p is $\eta_p = \frac{\eta_0}{(1+\mu q)^\nu}$, where q is the training progress linearly changing from 0 to 1, $\eta_0 = 0.01, \mu = 10, \nu = 0.75$.

Neural network architectures: On digits datasets, we use CNN for our generator and 1 FC layer for our classifier in both adaptation scenarios. or Office-Home dataset, our generator is a ResNet50 pre-trained on ImageNet excluding the last FC layer, which is our classifier.

Generator architecture was used for the SVHN dataset:
$z \in \mathbb{R}^{32\times32\times3} \to Conv_{32} \to BatchNorm \to ReLU \to Conv_{32} \to BatchNorm \to ReLU \to MaxPool2D \to Conv_{64} \to BatchNorm \to ReLU \to Conv_{64} \to BatchNorm \to ReLU \to MaxPool2D \to Conv_{128} \to BatchNorm \to ReLU \to Conv_{128} \to BatchNorm \to ReLU \to MaxPool2D \to Sigmoid \to FC_{128}$

Generator architecture was used for the USPS dataset:
$z \in \mathbb{R}^{28\times28\times3} \to Conv_{32} \to BatchNorm \to ReLU \to MaxPool2D \to Conv_{64} \to BatchNorm \to ReLU \to Conv_{128} \to BatchNorm \to ReLU \to MaxPool2D \to Sigmoid \to FC_{128}$

Classifier architecture was used for both SVHN and USPS datasets:
$z \in \mathbb{R}^{128} \to FC_{10}$

Classifier architecture was used for the Office-Home dataset:
$z \in \mathbb{R}^{512} \to FC_{65}$

Classifier architecture was used for the VisDA dataset:
$z \in \mathbb{R}^{512} \to FC_{12}$

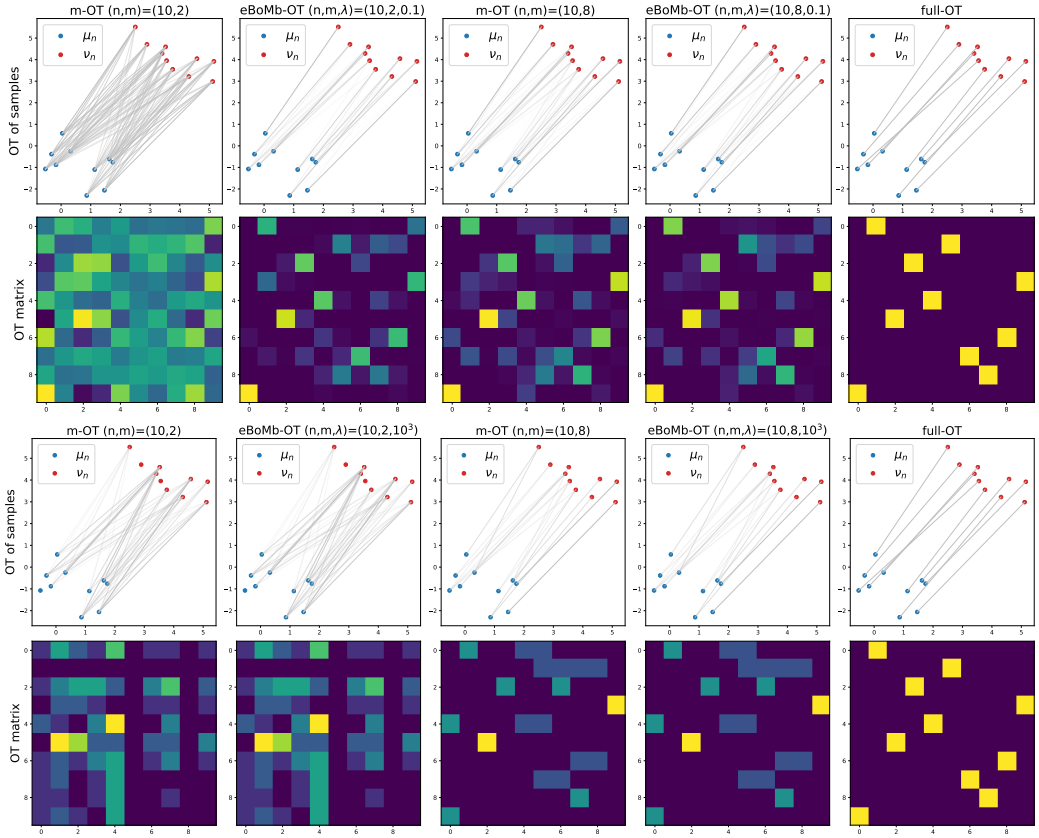

Figure 15. Visualization of the m-OT's transportation plan and the entropic regularized BoMb-OT's transportation plan between two 2D empirical distributions with 10 samples, and the interpolation property of the entropic BoMb-OT (eBoMb-OT). The first four rows provide the result for the entropic BoMb-OT ($\lambda \in \{0.1, 10^3\}$), and $k = 40$.

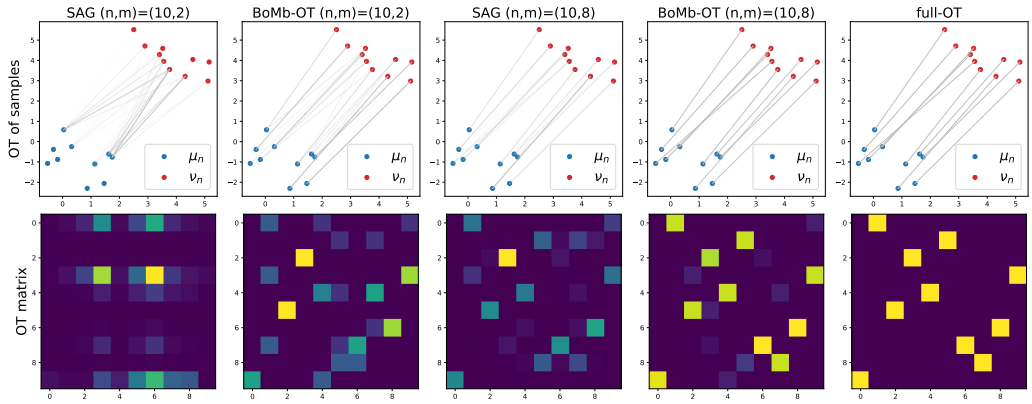

Figure 16. Visualization of the transportation plan of the BoMb-OT and the stochastic averaged gradient (SAG) method between two 2D empirical distributions with 10 samples. The number of mini-batch for the BoMb-OT is set to 30. The learning rate for SAG is the best in $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10\}$ and the entropic regularization of SAG is the best in $\{0.01, 0.05, 0.1, 0.5, 1, 5, 10\}$.

E.3. Gradient flow

For the implementation of the gradient flow, we use the geomloss library (Feydy et al., 2019). The learning rate is set to 0.001. For the autoencoder in CelebA experiments, we use the repo in https://github.com/rasbt/deeplearning-models/blob/master/pytorch_ipynb/autoencoder/ae-conv-nneighbor-celeba.ipynb for the pre-trained autoencoder.

E.4. Computational infrastructure

All deep learning experiments are done on a GTX 1080 Ti. Other experiments are done on a MacBook Pro 11inc M1.