# Scalable Computations of Wasserstein Barycenter via Input Convex Neural Networks

**Jiaojiao Fan** [1] **Amirhossein Taghvaei** [2] **Yongxin Chen** [1]

## Abstract

Wasserstein Barycenter is a principled approach to represent the weighted mean of a given set of probability distributions, utilizing the geometry induced by optimal transport. In this work, we present a novel scalable algorithm to approximate the Wasserstein Barycenters aiming at high-dimensional applications in machine learning. Our proposed algorithm is based on the Kantorovich dual formulation of the Wasserstein-2 distance as well as a recent neural network architecture, input convex neural network, that is known to parametrize convex functions. The distinguishing features of our method are: i) it only requires samples from the marginal distributions; ii) unlike the existing approaches, it represents the Barycenter with a generative model and can thus generate infinite samples from the barycenter without querying the marginal distributions; iii) it works similar to Generative Adversarial Model in one marginal case. We demonstrate the efficacy of our algorithm by comparing it with the state-of-art methods in multiple experiments. [1]

## 1. Introduction

The Wasserstein barycenter is concerned with the (weighted) average of multiple given probability distributions. It is based on the natural geometry over the space of distributions induced by optimal transport (Villani, 2003) theory and serves as a counterpart of arithmetic mean/average for data of distribution-type. Compared to other methods, Wasserstein barycenter provides a principled approach to average probability distributions, fully utilizing the underlying geometric structure of the data (Agueh & Carlier, 2011). During the past few years, it has found applications in several machine learning problems. For instance, in sensor fusion, Wasserstein barycenter is used to merge/average datasets collected from multiple sensors to generate a single collective result (Elvander et al., 2018). The advantage of Wasserstein barycenter is its ability to preserves the modality of the different datasets, a highly desirable property in practice (Jiang et al., 2012). Wasserstein Barycenter has also been observed to be effective in removing batch effects of the sensor measurements (Yang & Tabak, 2019). It has also found application in large scale Bayesian inference for averaging the results from Markov chain Monte Carlo (MCMC) Bayesian inference carried out over subsets of the observations (Srivastava et al., 2015; Staib et al., 2017; Srivastava et al., 2018). It has also been useful in image processing for texture mixing (Rabin et al., 2011) and shape interpolation (Solomon et al., 2015).

The bottleneck of utilizing Wasserstein barycenter in machine learning applications remains to be computational. Indeed, when the data is discrete, namely, the given probability distributions are over discrete space (e.g., grid), the Wasserstein barycenter problem can be solved using linear programming (Anderes et al., 2016). This has been greatly accelerated by introducing an entropy term (Cuturi & Doucet, 2014, Algorithm 1) (Solomon et al., 2015) as in Sinkhorn algorithm (Cuturi, 2013). However, these methods are not suitable for many machine learning applications involving distributions over continuous space. First of all, it requires discretization of the distribution to implement these methods and thus doesn't scale to high dimensional settings. Secondly, in some applications such as MCMC Bayesian inference (Andrieu et al., 2003; Srivastava et al., 2018) the explicit formulas of the distributions are not accessible, which precludes these discretization-based algorithms. When the support atoms are free to move, there are algorithms that interchangeably optimize the support weights and locations (Cuturi & Doucet, 2014, Algorithm 2) (Luise et al., 2019; Altschuler & Boix-Adsera, 2021) which can also be formulated in stochastic optimization framework (Claici et al., 2018). But these free-support methods become computationally highly expensive when the number of support points is large.

[1]Georgia Institute of Technology [2]University of California, Irvine. Correspondence to: Yongxin Chen <yongchen@gatech.edu>.

*Proceedings of the $38^{th}$ International Conference on Machine Learning*, PMLR 139, 2021. 

[1]Download Code at https://github.com/sbyebss/Scalable-Wasserstein-Barycenter

**Contribution:** We propose a computationally efficient and scalable algorithm for estimating the Wasserstein barycenter of probability distributions over continuous spaces. Our method is based on a Kantorovich-type dual characterization of the Wasserstein barycenter, which involves optimization over convex functions, and the recently introduced input convex neural networks (ICNN) (Amos et al., 2017; Chen et al., 2018b), that provides powerful representation of convex functions. Remarkably, in our framework, the (weighted) barycenter is represented by a generator network (Goodfellow et al., 2014; Arjovsky et al., 2017), that allows characterization of continuous distributions and fast and unlimited sampling from barycenter. We prove the consistency of our formulation in Proposition 1, and demonstrate its performance and its scaling properties in truly high-dimensional setting through extensive evaluations over various benchmark experiments including synthetic and real data-set and provide comparisons with several state-of-art algorithms: Korotin et al. (2021b), Li et al. (2020), Cuturi & Doucet (2014). Our experiments reveal significant improvement in estimating the barycenter in high-dimensional setting compared to most existing algorithms. We also showcase the the ability of our method to perform as a generative adversarial network (GAN) in the one marginal case and propose a heuristic extension to learn barycenter of arbitrary weights through a single training process.

**Related work:** Our proposed algorithm is closely related to continuous Wasserstein barycenter proposed by Li et al. (2020) and Korotin et al. (2021b). Similar to our approach, both of them are based on dual formulation of the Wasserstein barycenter problem and representing the potential functions with neural networks. However, the approach in Li et al. (2020) does not restrict the potential functions to be convex. Instead, an entropic or $l_2$ regularization term is added to ensure that the optimal potential functions are approximately convex. The addition of the regularization term introduces undesirable bias error which becomes severe in high-dimensional problems as shown in Figure 5-6 and also reported in (Korotin et al., 2021b, Table1-4) (Li et al., 2020, Section 5). The approach in Korotin et al. (2021b) restricts the potentials to be convex using ICNN, however, it includes a cycle regularizer term to ensure the potential functions are dual conjugate. Their formulation also involves a congruence regularizer to guarantee that the optimal potential functions are consistent with the true barycenter. The congruence regularizer requires selection of a priory probability distribution that is bounded below by the true barycenter, which is a non-trivial task. Moreover, the addition of the regularization terms distorts the nice optimization landscape of the original problem. The problem may become non-convex even for the simple setting of restriction to quadratic functions (see Sec. G in supplementary material). In contrast, our formulation does not involve additional regularization terms and retains the optimization landscape of the original problem. Moreover, a distinct feature of our algorithm is representing the barycenter using a generative model which allows a low-dimensional representation of the barycenter and access to infinitely many samples, while both of these methods represent the barycenter using the Monge maps from the marginals, and limits the number of samples to the number available from marginal distributions.

Earlier stochastic Wasserstein barycenter method (Claici et al., 2018) also aims at calculating barycenters for continuous distributions using samples. However, they adopt a semi-discrete approach that models the barycenter with a finite set of points. That is, even though the marginals are continuous, the barycenter is discrete. Several other sample-based algorithms (Staib et al., 2017; Kuang & Tabak, 2019; Mi et al., 2020) are also of semi-discrete-type. Most other Wasserstein barycenter algorithms are for discrete distributions and require discretization if applied to continuous distributions. An incomplete list includes (Cuturi & Doucet, 2014; Benamou et al., 2015; Solomon et al., 2015).

The subject of this work is also related to the vast amount of literature on estimating the optimal transport map and Wasserstein distance (see (Peyré et al., 2019) for a complete list). Closely related to this paper are the recent works that aim to extend the optimal transport map estimation to large-scale machine learning settings (Genevay et al., 2016; Seguy et al., 2017; Liu et al., 2018; Chen et al., 2018a; Leygonie et al., 2019; Xie et al., 2019). In particular, our algorithm is inspired by the recent advances in estimation of optimal transport map and Wasserstein-2 distance using ICNNs (Taghvaei & Jalali, 2019; Makkuva et al., 2020; Korotin et al., 2021a).

## 2. Background

### 2.1. Optimal transport and Wasserstein distance

Given two probability distributions $\nu, \mu$ over Euclidean space $\mathbb{R}^n$ with finite second moments, the optimal transport (Villani, 2003) (OT) problem with quadratic unit cost seeks an optimal joint distribution of $\nu, \mu$ that minimizes the total transport cost. More specifically, it is formulated as $W_2^2(\nu, \mu) := \min_{\pi \in \Pi(\nu,\mu)} \int_{\mathbb{R}^n \times \mathbb{R}^n} \|x - y\|^2 d\pi(x, y)$, where $\Pi(\nu, \mu)$ denotes the set of all joint distributions of $\nu$ and $\mu$. The square-root of the minimum transport cost defines the celebrated Wasserstein-2 distance $W_2$, which is known to enjoy many nice geometrical properties compared to other distances for distributions (Ambrosio et al., 2008).

The Kantorovich dual of the OT problem reads

$$\frac{1}{2}W_2^2(\nu, \mu) = \sup_{(\phi,\psi)\in\Phi} \mathbb{E}_\nu[\phi(X)] + \mathbb{E}_\mu[\psi(Y)], \quad (1)$$

where $\Phi := \{(\phi, \psi) \in L^1(\nu) \times L^1(\mu)\,;\, \phi(x) + \psi(y) \leq$

$\frac{1}{2}\|x-y\|^2,\ \ \forall x, y\}$. Let $f(x) = \|x\|^2/2 - \phi(x)$, then (1) can be rewritten as

$$\frac{1}{2}W_2^2(\nu,\mu) = C_{\nu,\mu} - \inf_{f\in\mathbf{CVX}}\{\mathbb{E}_\nu[f(X)] + \mathbb{E}_\mu[f^*(Y)]\} \quad (2)$$

where **CVX** stands for the set of convex functions, $C_{\nu,\mu} := (1/2)\{\mathbb{E}_\nu[\|X\|^2] + \mathbb{E}_\mu[\|Y\|^2]\}$, and the $f^*$ is the convex conjugate (Rockafellar, 1970) function of $f$. The formulation (2) is known as the semi-dual formulation of OT. The **CVX** condition restricts the search space for $f$ which becomes handy for design of optimization algorithms.

**Remark 1** *When both of the marginal distributions have densities, Brenier' Theorem gives that $\nabla f^*$ is the optimal transport map from $\mu$ to $\nu$ (Villani, 2003) and $\nabla f$ is the optimal map from $\nu$ to $\mu$.*

### 2.2. Wasserstein Barycenter

Wasserstein barycenter is OT-based average of probability distributions. Given a set of probability distributions $\mu_i$, $i = 1, 2, \ldots, N$ and a weight vector $a \in \mathbb{R}^N$ ($a_i \geq 0$, $i = 1, 2, \ldots, N$ and $\sum_{i=1}^N a_i = 1$), the associated Wasserstein barycenter is defined as the minimizer of

$$\min_\nu \sum_{i=1}^N a_i W_2^2(\nu, \mu_i). \quad (3)$$

The barycenter problem (3) can be reformulated as a linear programming (Agueh & Carlier, 2011). However, the linear programming-base algorithms don't scale well for high dimensional problems. A special case that can be solved efficiently is when the marginal distributions $\{\mu_i\}$ are Gaussian. Denote the mean and covariance of $\mu_i$ as $m_i$ and $\Sigma_i$ respectively, then their Wasserstein barycenter is a Gaussian distribution with mean being $m = \sum_{i=1}^N a_i m_i$ and covariance $\Sigma$ being the unique solution to the fixed-point equation $\Sigma = \sum_{i=1}^N a_i(\Sigma^{1/2}\Sigma_i\Sigma^{1/2})^{1/2}$. In Álvarez-Esteban et al. (2016), a simple however efficient algorithm was proposed to solve for $\Sigma$.

### 2.3. Input Convex Neural Network

Input Convex Neural Network (ICNN) is a type of deep neural networks architecture that characterize convex functions (Amos et al., 2017). A fully ICNN (FICNN) leads to a function that is convex with respect to all inputs.

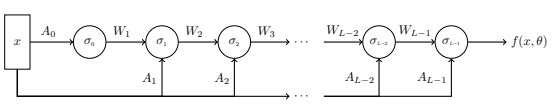



Figure 1: Fully input convex neural network (FICNN)

The FICNN architecture is shown in Fig. 1. It is a $L$-layer feedforward neural network propagating following, for $l = 0, 1, \ldots, L-1$

$$z_{l+1} = \sigma_l(W_l z_l + A_l x + b_l), \quad (4)$$

where $\{W_l\}, \{A_l\}$ are weight matrices (with the convention that $W_0 = 0$), $\{b_l\}$ are the bias terms, and $\sigma_l$ denotes the entry-wise activation function at the layer $l$. Denote the total set of parameters by $\theta = (\{W_l\}, \{A_l\}, \{b_l\})$, then this network defines a map from input $x$ to $f(x;\theta) = z_L$. This map $f(x;\theta)$ is convex in $x$ provided 1) $W_{1:L-1}$ are non-negative; 2) $\sigma_{0:L-1}$ are convex; 3) $\sigma_{1:L-1}$ are non-decreasing (Makkuva et al., 2020). We remark that FICNN has the ability to approximate any convex function over a compact domain with a desired accuracy (Chen et al., 2018b), which makes FICNN an ideal candidate for modeling convex functions.

## 3. Methods and algorithms

We study the Wasserstein barycenter problem (3) for a given set of marginal distributions $\{\mu_i;\ i = 1, \ldots, N\}$. We consider the setting where the analytic forms of the marginals are not available. Instead, we only have access to independent samples from them. It can be either the cases where a fix set of samples is provided as in supervised learning, or the cases where one can keep sampling from the marginals like in the MCMC Bayesian (Srivastava et al., 2018). Our goal is to recover the true continuous Barycenter $\nu$.

### 3.1. Deriving the dual problem over convex functions

For a fixed $\nu$, the objective function of (3) is simply a (scaled) summation of the Wasserstein cost between $\nu$ and $\mu_i$. Thus, we utilize the semi-dual formulation (2) of OT to evaluate the objective function of (3). However, convex conjugate function $f^*$ is not available explicitly in most of applications, thus we characterize it as

$$f^*(y) = \sup_{g\in\mathbf{CVX}} \langle y, \nabla g(y)\rangle - f(\nabla g(y)) \quad (5)$$

with the maximum being achieved at $g = f^*$, the semi-dual formulation (2) can be rewritten as

$$\frac{1}{2}W_2^2(\nu,\mu) = \sup_{f\in\mathbf{CVX}} \inf_{g\in\mathbf{CVX}} \mathcal{V}_{\nu,\mu}(f,g) + C_{\nu,\mu}, \quad (6)$$

where $\mathcal{V}_{\nu,\mu}(f,g)$ is a functional of $f$ and $g$ defined as

$$\mathcal{V}_{\nu,\mu}(f,g) = -\mathbb{E}_\nu[f(X)] - \mathbb{E}_\mu[\langle Y, \nabla g(Y)\rangle - f(\nabla g(Y))]. \quad (7)$$

This formulation (6) has been utilized in conjugation with FICNN to solve OT problem in (Makkuva et al., 2020) and proved to be advantageous.

Plugging (6) into the Wasserstein barycenter problem (3), we obtain the following reformulation

$$\min_\nu \sum_{i=1}^N a_i \left\{ \sup_{f_i\in\mathbf{CVX}} \inf_{g_i\in\mathbf{CVX}} \mathcal{V}_{\nu,\mu_i}(f_i, g_i) + C_{\nu,\mu_i} \right\}. \quad (8)$$

Note that we have used different functions $(f_i, g_i)$ to estimate $W_2^2(\nu, \mu_i)$. The first minimization is over all the possible probability distributions to search for the Wasserstein barycenter. This min-max-min formulation enjoys the following property, whose proof is in the supplementary material.

**Proposition 1** *When all the marginal distributions $\mu_i$ are absolutely continuous with respect to the Lebesgue measure, the unique Wasserstein barycenter $\nu^\star$ of them solves (8). Moreover, the corresponding optimal $f_i^\star$ is the optimal potential in (2) associated with marginals $\nu^\star$ and $\mu_i$.*

**Remark 2** *Obtaining convergence rate for first-order optimization algorithms solving (8) is challenging even in the ideal setting that the optimization is carried out in the space of probability distributions. The difficulty arises because of the optimization over $\nu$. While the inner optimization over $f_i$ and $g_i$ are concave and convex respectively, the optimization over $\nu$ is not convex. Precisely, it is not geodesically convex on the space of probability distributions equipped with Wasserstein-2 metric (Ambrosio et al., 2008). However, it is possible to obtain guarantees in a restricted setting by establishing a Polyak- Lojasiewicz type inequality. In particular, assuming all $\mu_i$ are Gaussian with positive-definite covariance matrices, the gradient-descent algorithm admits a linear convergence rate (Chewi et al., 2020).*

### 3.2. Solving the barycenter problem

Consider the Wasserstein barycenter problem for a fixed weight vector $a$. Following (Makkuva et al., 2020) we use FICNN architecture to represent convex functions $f_i$ and $g_i$. We now use a generator $h$ to model the distribution $\nu$, by transforming samples from a simple distribution $\eta$ (e.g., Gaussian, uniform) to a complicated distribution, thereby we recover a continuous Barycenter distribution. Thus, using this network parametrization and discarding constant terms, we arrive at the following optimization problem

$$\min_{h} \sup_{f_i \in \textbf{FICNN}} \inf_{g_i \in \textbf{FICNN}} \frac{1}{2}\mathbb{E}_\eta[\|h(Z)\|^2] + \sum_{i=1}^{N} a_i \overline{\mathcal{V}}_{\eta,\mu_i}(f_i, g_i), \tag{9}$$

where $\overline{\mathcal{V}}_{\eta,\mu_i}(f, g)$ is defined as

$$-\mathbb{E}_\eta[f_i(h(Z))] - \mathbb{E}_{\mu_i}[\langle Y^i, \nabla g_i(Y^i)\rangle - f_i(\nabla g_i(Y^i))].$$

We propose Neural Wasserstein Barycenter (NWB) algorithm (Algorithm 1) to solve this three-loop min-max-min problem by alternatively updating $h$, $f_i$ and $g_i$ using stochastic optimization algorithms. This pipeline is illustrated by the block diagram (Figure 2). We remark that the objective function in (9) can be estimated using samples from $\mu_i, \eta$. Thus, we just need access to the samples generated by the marginal distributions $\mu_i$ instead of their analytic form to compute their Wasserstein barycenter. In practice, we found it more effective to replace the convexity constraints for $g_i$ with a convexity penalty, that is, the negative values of the weight matrices $W_l$ in FICNN (4).

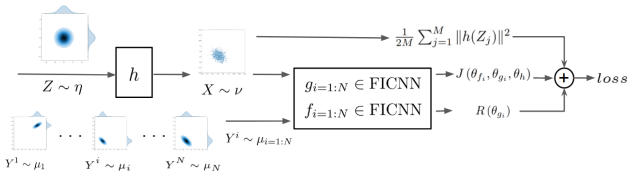



Figure 2: Block diagram for our neural Wasserstein barycenter (NWB) algorithm

Denoting the parameters of $h, f_i, g_i$ by $\theta_h, \theta_{f_i}, \theta_{g_i}$ respectively and the batch size by $M$, we arrive at the batch estimation of the objective

$$\sum_{i=1}^{N} a_i \left[J(\theta_{f_i}, \theta_{g_i}, \theta_h) + R\left(\theta_{g_i}\right)\right] + \frac{1}{2M}\sum_{j=1}^{M} ||h(Z_j)||^2, \tag{10}$$

where $J\left(\theta_{f_i}, \theta_{g_i}, \theta_h\right)$ represents

$$\frac{1}{M}\sum_{j=1}^{M} f_i\left(\nabla g_i\left(Y_j^i\right)\right) - \left\langle Y_j^i, \nabla g_i\left(Y_j^i\right)\right\rangle - f_i\left(h(Z_j)\right),$$

$R\left(\theta_{g_i}\right) = \lambda \sum_{W_l \in \theta_{g_i}} \|\max\left(-W_l, 0\right)\|_F^2$, $Y_j^i$ represents the $j^{th}$ sample generated by $\mu_i$, $\{Z_j\}$ are samples from $\eta$, and $\lambda > 0$ is a hyper-parameter weighing the intensity of regularization.

Algorithm 1 can be extended to obtain the barycenters for all weights in one shot. This extension is included in Sec. B of the supplementary material.

**Remark 3** *It is tempting to combine the two minimization steps over $h$ and $g_i$ into one and reduce (9) into a min-max saddle point problem. The resulting algorithm only alternates between $f_i$ updates and $h, g_i$ updates instead of the three-way alternating in Algorithm 1. However, in our implementations, we observed that this strategy is unstable.*

**Computation complexity** For our algorithm, as well as the algorithms recently proposed in (Korotin et al., 2021b; Li et al., 2020), the computational complexity per iteration scales with $O(NMp)$ where $N$ is the number of marginals, $M$ is the batch-size, and $p$ is the size of the network (the size of network scales almost linearly with dimension $d$). This should be compared with $O(NMK)$ for Claici et al. (2018) where $K$ is the size of the support for barycenter, and $O(NnK)$ for Cuturi & Doucet (2014) where $n$ is the number of samples of the marginals. Although the size of the network is large, our approach is favored for large scale

**Algorithm 1** Neural Wasserstein Barycenter (NWB)

**Input:** Marginal dist. $\mu_{1:N}$, Generator dist. $\eta$, Batch size $M$
**for** $k_3 = 1, \dots, K_3$ **do**
  Sample batch $\{Z_j\}_{j=1}^M \sim \eta$
  Sample batch $\left\{Y_j^i\right\}_{j=1}^M \sim \mu_i$ for all $i = 1, \cdots, N$
  **for** $k_2 = 1, \dots, K_2$ **do**
    **for** $k_1 = 1, \dots, K_1$ **do**
      Update all $\theta_{g_i}$ to decrease (10)
    **end for**
    Update all $\theta_{f_i}$ to increase (10)
    Clip: $W_l = \max(W_l, 0)$ for all $\theta_{f_i}$
  **end for**
  Update $\theta_h$ to decrease (10)
**end for**

problems where the number of samples $n$ and the dimension $d$ are large (the number of atoms required to approximate a density scales exponentially with dimension).

### 3.3. Recovering the barycenter

Once Algorithm 1 converges, there are two distinct approaches to recover the Wasserstein barycenter: one through $h$ and one through $g_i$.

**Generative model $h_\sharp\eta$:** In our problem formulation (9), the barycenter center is modeled by $h(Z)$ where $Z$ is sampled from a simple distribution $\eta$. That is, the barycenter $\nu$ is the pushforward of $\eta$ through the map $h$, denoted by $h_\sharp\eta$. Once the optimal $h$ is obtained, we can easily sample from the barycenter by sampling $Z_j$ from $\eta$ and apply the map $h(Z_j)$.

**Pushforward map $\nabla g_i \sharp \mu_i$:** An alternative method to recover the barycenter is based on the fact, once Algorithm 1 converges, the pair $(f_i, g_i)$ solves the OT problem (6) between the barycenter $\nu$ and the marginal $\mu_i$. As mentioned in Remark 1, $\nabla f^*$ is the optimal map from marginal distribution to the barycenter. Moreover, the optimal $g_i$ is achieved at $g = f^*$. Hence, the pushforward of $\mu_i$ through map $\nabla g_i$ is the barycenter. Thus, to sample from the barycenter, we can sample $Y_j^i$ from a marginal and then apply map $\nabla g_i(Y_j^i)$. Note that this approach cannot generate more samples than those are already available in the marginals.

### 3.4. Barycenter serving as GAN

In case where there is only one marginal distribution, that is, $N = 1$ in (9), Algorithm 1 can be viewed as a type of generative adversarial network (GAN). More specifically, when $N = 1$, the barycener $\nu$ coincides with the marginal distribution $\mu_1$. Given samples $\{Y_1^j\}$ from the marginal $\mu_1$, Algorithm 1 produces a generative model $h(Z)$ whose distribution matches the marginal $\mu_1$. Note that one can easily sample using $h(Z)$ and get samples that do not exist in the training data $\{Y_1^j\}$.

In fact, when $N = 1$, Algorithm 1 works very much like a Wasserstein Generative Adversarial Network (WGAN) which leverages the Wasserstein distance to distinguish fake and real samples in GAN. The original WGAN (Arjovsky et al., 2017) is based on the dual formulation for the Wasserstein-1 distance $W_1$. A heuristic weight clipping (Arjovsky et al., 2017) technique is used to enforce the Lipschitz condition on the potential function in the dual formulation. The WGAN was later improved in WGAN-GP (Gulrajani et al., 2017) via adding a gradient penalty term to promote the Lipschitz condition. From this point of view, Algorithm 1 provides an alternative way to train the generative model with Wasserstein-2 metric (c.f. Leygonie et al. (2019); Korotin et al. (2021a); Salimans et al. (2018); Genevay et al. (2018)).

## 4. Experiments

In Section 4.1, we present numerical experiments on 2D/3D datasets which serve as proof of concept and qualitatively illustrate the performance of our approach. In Section 4.2, we numerically study the effects of dimensionality and demonstrate the scalability of our algorithms to high-dimensional problems. In Section 4.3 and 4.4, we apply our algorithm in tasks such as Bayesian inference with large scale dataset and color transfer. In Section 4.5, we illustrate the ability of our algorithm to serve as a generative model. The implementation details and further experiments are included in the supplementary materials.

For comparison, we choose the following state of the art algorithms: (i) fast free-support Wasserstein barycenter (CDWB) (Cuturi & Doucet, 2014, Section 4.4); (ii) continuous Wasserstein barycenter without minimax optimization (CWB) (Korotin et al., 2021b); (iii) continuous regularized Wasserstein barycenter (CRWB) (Li et al., 2020). CWB and CRWB involve optimization over $N$ pairs of potentials $\{f_i, g_i\}$ as in NWB, and recover barycenter through $\nabla g_i \sharp \mu_i$. The implementations of these algorithms are based on published code associated with the papers.

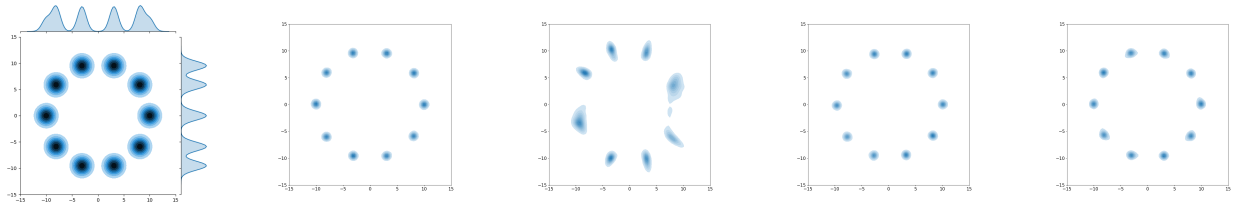
To evaluate the performance of these algorithms, we use the Bures-Wasserstein UVP (Korotin et al., 2021b, Section 5)

$$\text{BW}_2^2\text{–UVP}\left(\nu, \tilde{\nu}\right) \stackrel{\text{def}}{=} 100\frac{\text{BW}_2^2\left(\nu, \tilde{\nu}\right)}{\frac{1}{2}\text{Var}(\tilde{\nu})}\%, \qquad (11)$$

where $\text{BW}_2^2\left(\nu, \tilde{\nu}\right)$ equals

$$\frac{1}{2}\|m_\nu - m_{\tilde{\nu}}\|^2 + \left[\frac{1}{2}\text{Tr}\Sigma_\nu + \frac{1}{2}\text{Tr}\Sigma_{\tilde{\nu}} - \text{Tr}\left(\Sigma_\nu^{\frac{1}{2}}\Sigma_{\tilde{\nu}}\Sigma_\nu^{\frac{1}{2}}\right)^{\frac{1}{2}}\right].$$

Here $\nu$ is the estimated barycenter, $\tilde{\nu}$ is the exact barycenter,



(a) marginal $\mu_1$ (b) NWB (ours) (c) WGAN (d) WGAN-GP (e) W2GN

Figure 3: performance of our algorithm as GAN in single marginal case ($N = 1$): learning Gaussian mixture

and $m_\nu, \Sigma_\nu$ are the mean and the convariance of the distribution $\nu$. For barycenter given by pushforward $\nabla g_i \sharp \mu_i$, we report the weighted average of $\text{BW}_2^2$–UVP scores from $N$ marginal distributions: $\sum_{i=1}^N a_i \text{BW}_2^2\text{–UVP}\left(\nabla g_i \sharp \mu_i, \tilde{\nu}\right)$.

**Hyper-parameter choice** We choose a neural network architecture of $3 \sim 4$ hidden layers of size $1 \sim 2$ times of input dimension in high dim cases and size $16 \sim 32$ in 2D/3D cases, with **PReLU** activation function. We observed that the performance is not sensitive to number of hidden layers, but sensitive to the choice of activation function (ReLU and leaky ReLU do not perform as well as PReLU).

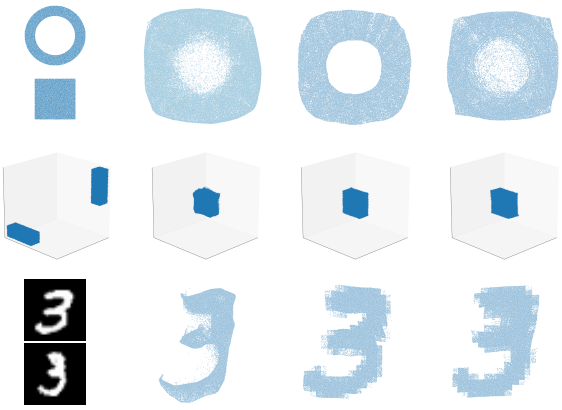
**Training time:** The training time for our method is almost once to twice longer than of CRWB and CWB due to the inner optimization cycles. The time consumption for CDWB is shorter for small number of samples, as it does not involve training neural networks.

### 4.1. Learning the Wasserstein Barycenter in 2D and 3D

The qualitative performance of our algorithm in three benchmark examples is depicted in Figure 4. Each example is represented in a row. The first column contains the marginal distributions, and the second and third column contains the learned barycenter through $h\sharp\eta$ and $\nabla g_i \sharp \mu_i$ respectively. It is observed that both representations learn the barycenter qualitatively well, however, representation through $\nabla g_i \sharp \mu_i$ inherits the geometrical properties of the marginal distributions, highlighted with sharp boundaries in the first and second row and pixelated image in the third row. For comparison with CRWB, see Li et al. (2020, Figure 1).

### 4.2. Scalability with the dimension

**Gaussian:** We study the performance of our proposed algorithm in learning the barycenter of three Gaussian marginal distributions as dimension grows. The Gaussian marginal distributions have zero mean and a random non-diagonal covariance matrix whose conditional number is less than 10. The exact barycenter of Gaussian distributions is available to serve as the baseline in evaluating the Bures-Wasserstein UVP error criteria (11). The results are displayed in Figure 5. It is observed that the estimation error of NWB and CWB



marginals (a) $h\sharp\eta$ (b) $\nabla g_1 \sharp \mu_1$ (c) $\nabla g_2 \sharp \mu_2$

Figure 4: Qualitative results of our algorithm (NWB) in 2D and 3D settings. The first column contains the marginals, and the second and third columns contain the barycenter generated through $h\sharp\eta$ and $\nabla g_i \sharp \mu_i$ respectively. From top to bottom, the marginal distributions are uniformly supported on a ring and a square; two blocks; and white areas in digit images.

exhibit a slow rate of growth with respect to the dimension compared with CDWB and CRWB. The NWB and CWB algorithms are quite comparable in performance, but still different with respect to the optimization landscape. The optimization landscape of CWB is sensitive to the choice of regularization parameters (see Appendix G). We conjecture that the observed error curve of CWB in Figure 5, is due to the effect of regularization terms in distorting the optimization landscape which becomes more severe as dimension grows. Note that the error curve for CRWB contains irregularities and high variance that are probably due to the regularization term.

**MNIST:** To further investigate the performance of our algorithm in high dimension setting with real dataset, we use the MNIST data set. We consider the task of learning the barycenter of two marginal distributions. The first marginal $\mu_1$ is an empirical distribution of digit 0 samples and the second marginal $\mu_2$ is of digit 1. Each image has $28 \times 28$ pixels, yielding a 784-dimensional problem. The result of learning the barycenter is depicted in Figure 6. Both our algorithm NWB and CWB give reasonable results, with

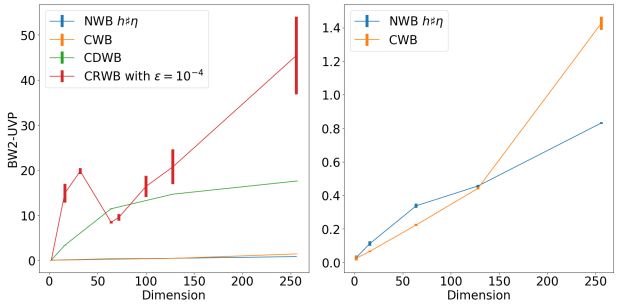


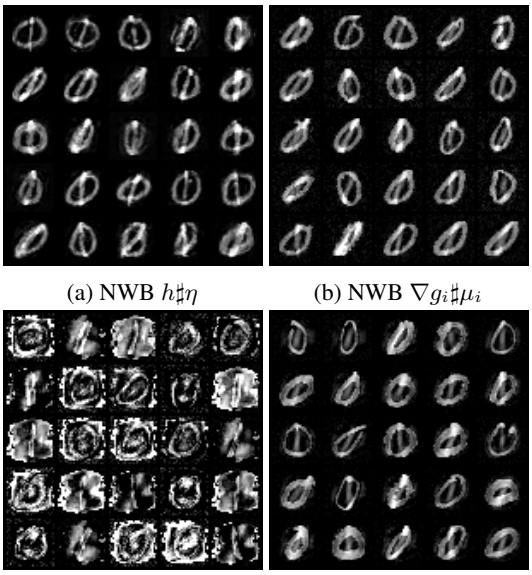

Figure 5: Numerical result for scalability of the error with dimension for estimating barycenter of Gaussian distributions. The error criteria is (11). We generate $10^4$ samples from barycenter for NWB, CWB, CRWB algorithms and 1500 samples for CDWB (the maximum number it can generate). The plot on the left includes the results for all four methods and the plot on the right highlights the detailed difference between CWB and NWB.

slightly sharper boundaries in NWB, whereas CRWB does not perform well in this high dimension setting. Note that the images in panel (a) are genuinely new samples generated using the trained generator, while the images in other panels are pushforward of marginal samples.

In order to demonstrate the inner-workings of our algorithm and its ability to learn the structure of barycenter, we implement the following experiment. We generate fresh samples from the barycenter using the generator $h(Z)$, where $Z \sim \mathcal{N}(\mathbf{0}, I)$, and push-forward it through the maps $\nabla f_1(h(Z))$ and $\nabla f_2(h(Z))$. It is expected that through this procedure, we would recover the digits 0 and 1, because $\nabla f_i$ represent the optimal transport map from Barycenter to the marginal $\mu_i$ (see Remark 1). The experimental result confirms our expectation as shown in Figure 7. This implies that our proposed framework can serve as a generative model, not only for barycenter distribution, but also for the marginal distributions, by taking a random Gaussian random variable as input and output samples from marginal distributions.

### 4.3. Subset posterior aggregation

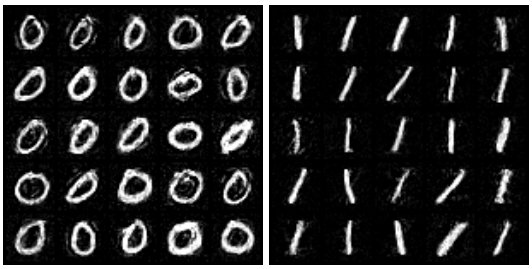
MCMC Bayesian inference is often carried out on splitted datasets in big data setting. However, the subset posterior distributions need to be merged into a single posterior to reflect the entire dataset property. This subset posterior aggregation scheme has been shown as an advantageous substitute to the full posterior (Srivastava et al., 2015) (Srivastava et al., 2018). The barycenter of subset posteriors is proved to converge to the full data posterior (Srivastava et al., 2018). Similar to (Li et al., 2020), we consider the Poisson regression for predicting the hourly number of bike hires with predictors such as the season and the weather



(a) NWB $h\sharp\eta$ (b) NWB $\nabla g_i\sharp\mu_i$

(c) CRWB (d) CWB

Figure 6: Learning the barycenter of MNIST 0 and 1 digits (784-dimensional problem)



(a) backward to '0' (b) backward to '1'

Figure 7: Generating digit 0 and 1 from random input $Z \sim \mathcal{N}(\mathbf{0}, I)$ using our architecture with the map $\nabla f_i(h(Z))$.

conditions[2]. We consider the posterior on the 8-dimensional coefficients for the Poisson regression model. We randomly split the data into 5 equal-size subsets and simultaneously use the stochastic approximation trick (Minsker et al., 2014) to promise the subset posterior samples do not vary consistently from the full posterior in covariance. We obtain $10^5$ samples from each subset posterior using the PyMC3 library (Salvatier et al., 2016).

We use the full posterior samples as the ground truth and report the Bures-Wasserstein UVP to compare the estimated barycenter and the ground truth. The results are shown in Table 1. All methods approach the true barycenter well (UVP $< 1\%$) and the performance of NWB is better than or on par with existing algorithms.

[2] http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset

Table 1: Comparison of UVP for recovered barycenters in our subset posterior aggregation task

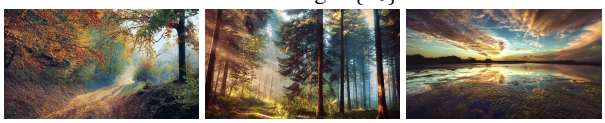
| METRIC | NWB $h\sharp\eta$ | CDWB | CWB | CRWB |
|---|---|---|---|---|
| $BW_2^2$–UVP, % | 0.06 | 0.26 | 0.07 | 1.67 |

### 4.4. Color palette averaging

Color transfer is a method to change the appearance of a source image according to the color pattern of a target image (Reinhard et al., 2001). Given several images, we can solve for Wasserstein barycenter to aggregate color palettes of images to achieve color transfer among them. Given an RGB image $\mathcal{I}$, its color palette is the empirical distribution $\mu(\mathcal{I}) = \sum_{k=1}^{K} \frac{1}{K}\delta_{p_k}$ where $\{p_k\}$ represents the pixels $\in [0,1]^3$ and $\delta_{p_k}$ is the Dirac distribution concentrated on $p_k$. In our example, each image contains $1980 \times 1024$ pixels, so the number of samples for each marginal distribution is more than 2 million. [3].

In Figure 8, the upper panel shows the original images $\{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3\}$, and the bottom panel shows pixel-wise "push-forward" images. Figure 9 shows the RGB cloud to visualize the color palettes of images. In Figure 8, the appearance of the pushforward images are different from the source images thanks to the color averaging: the leaves in the first picture become greener and darker, the sunbeams in the second picture become more red, and the sky in the last picture receives an orange color toning.

Source images $\{\mathcal{I}_i\}$



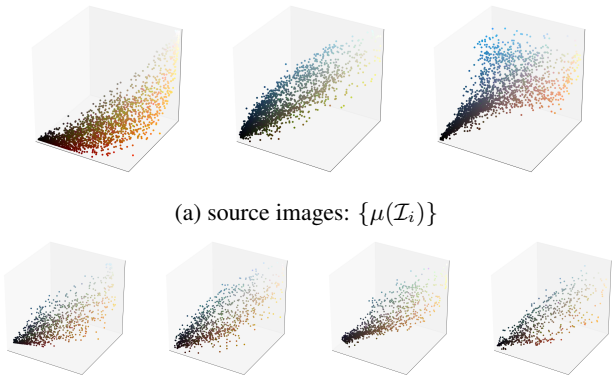
Pushforward images

Figure 8: Qualitative results by pixel-wise pushforward of the source images

### 4.5. Serving as a Generative Adversarial Model in the one marginal setting

We study the performance of our proposed algorithm in the case of one marginal distribution, where it behaves as a generative adversarial network using the $W_2$ metric. For comparison, we use WGAN (Arjovsky et al., 2017) and WGAN-GP (Gulrajani et al., 2017), which are based on $W_1$ metic, and W2GN (Korotin et al., 2021a), which is based on $W_2$ metric. Note that our goal is not to provide a competitive GAN algorithm, but to demonstrate the ability of our algorithm in performing as GAN.

[3]The pictures are downloaded from https://wallpaperaccess.com/



(a) source images: $\{\mu(\mathcal{I}_i)\}$

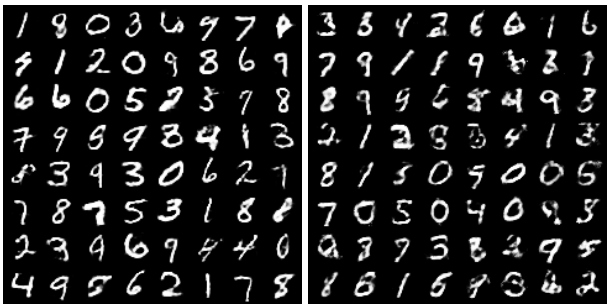
(b) $\nabla g_1 \sharp \mu(\mathcal{I}_1)$ (c) $\nabla g_2 \sharp \mu(\mathcal{I}_2)$ (d) $\nabla g_3 \sharp \mu(\mathcal{I}_3)$ (e) $h\sharp\eta$

Figure 9: Color palettes of source images and barycenter

We first consider an example of learning a Gaussian mixture model with 10 components shown in Figure 3. It can be seen that NWB avoids mode collapse. We then investigate the performance of our algorithm NWB in learning MNIST digits dataset (784 dim, 60000 sample size). From Figure 10, it is observed that our algorithm could output all the digits from 0 to 9 without mode collapse and the quality is on par with WGAN and WGAN-GP. W2GN seeks an optimal transport map from 784 dim standard Gaussian to MNIST; the generated digits are of poor quality. Note that in Korotin et al. (2021a) W2GN was tested on MNIST dataset but the optimal transport is addressed in the feature/latent space (see Section 5.2, Section C.7 in Korotin et al. (2021a)), which is of much lower dimension than the pixel space (784 dim).

## 5. Conclusion

During the last decade, many algorithms have been proposed for Wasserstein Barycenter estimation. A majority of these algorithms are designed for discrete setting (either discretization of space or discretization of distribution from samples). There are several algorithms that are designed for semi-discrete setting, in the sense that even though the marginal distributions are continuous, the barycenter computed from the algorithms is supported on finite points. More recently, two algorithms (Korotin et al., 2021b; Li et al., 2020) have been proposed to approximate the barycenter by learning the optimal transport maps from the marginal distributions to the barycenter using samples from the marginal distributions. Compared to all these existing Wasserstein Barycenter estimation algorithm, the NWB algorithm we develop is the only algorithm that gives a continuous representation of the barycenter through a generative model and is capable of generating infinitely many samples from the barycenter.



(a) NWB $h\sharp\eta$



(b) WGAN

(c) WGAN-GP

(d) W2GN

Figure 10: Performance of our algorithm as GAN in single marginal case: learning the MNIST digit dataset

## Acknowledgments

The authors would like to thank the anonymous reviewers for useful comments. JF and YC are supported in part by grants NSF CAREER ECCS-1942523, NSF CCF-2008513, and NSF CCF-1740776. The authors also thank Qinsheng Zhang for fruitful discussions.

## References

Agueh, M. and Carlier, G. Barycenters in the Wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2): 904–924, 2011.

Altschuler, J. M. and Boix-Adsera, E. Wasserstein barycenters can be computed in polynomial time in fixed dimension. *Journal of Machine Learning Research*, 22(44):1–19, 2021. URL http://jmlr.org/papers/v22/20-588.html.

Álvarez-Esteban, P. C., Del Barrio, E., Cuesta-Albertos, J., and Matrán, C. A fixed-point approach to barycenters in Wasserstein space. *Journal of Mathematical Analysis and Applications*, 441(2):744–762, 2016.

Ambrosio, L., Gigli, N., and Savaré, G. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

Amos, B., Xu, L., and Kolter, J. Z. Input convex neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 146–155. JMLR. org, 2017.

Anderes, E., Borgwardt, S., and Miller, J. Discrete wasserstein barycenters: Optimal transport for discrete data. *Mathematical Methods of Operations Research*, 84(2): 389–409, 2016.

Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. An introduction to MCMC for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Benamou, J.-D., Carlier, G., Cuturi, M., Nenna, L., and Peyré, G. Iterative Bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138, 2015.

Bonneel, N., Van De Panne, M., Paris, S., and Heidrich, W. Displacement interpolation using lagrangian mass transport. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, pp. 1–12, 2011.

Chen, Y., Georgiou, T. T., and Tannenbaum, A. Optimal transport for gaussian mixture models. *IEEE Access*, 7: 6269–6278, 2018a.

Chen, Y., Shi, Y., and Zhang, B. Optimal control via neural networks: A convex approach, 2018b.

Chewi, S., Maunu, T., Rigollet, P., and Stromme, A. J. Gradient descent algorithms for bures-wasserstein barycenters. *arXiv preprint arXiv:2001.01700*, 2020.

Claici, S., Chien, E., and Solomon, J. Stochastic Wasserstein barycenters. *arXiv preprint arXiv:1802.05757*, 2018.

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

Cuturi, M. and Doucet, A. Fast computation of Wasserstein barycenters. 2014.

Elvander, F., Haasler, I., Jakobsson, A., and Karlsson, J. Tracking and sensor fusion in direction of arrival estimation using optimal mass transport. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pp. 1617–1621. IEEE, 2018.

Flamary, R. and Courty, N. Pot python optimal transport library, 2017. URL https://pythonot.github.io/.

Genevay, A., Cuturi, M., Peyré, G., and Bach, F. Stochastic optimization for large-scale optimal transport. In *Advances in neural information processing systems*, pp. 3440–3448, 2016.

Genevay, A., Peyre, G., and Cuturi, M. Learning generative models with sinkhorn divergences. In Storkey, A. and Perez-Cruz, F. (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1608–1617. PMLR, 09–11 Apr 2018. URL http://proceedings.mlr.press/v84/genevay18a.html.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30: 5767–5777, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Jiang, X., Ning, L., and Georgiou, T. T. Distances and Riemannian metrics for multivariate spectral densities. *IEEE Transactions on Automatic Control*, 57(7):1723–1735, 2012.

Korotin, A. Wasserstein-2 generative networks, 2020. URL https://github.com/iamalexkorotin/Wasserstein2GenerativeNetworks.

Korotin, A., Egiazarian, V., Asadulaev, A., Safin, A., and Burnaev, E. Wasserstein-2 generative networks. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=bEoxzW_EXsa.

Korotin, A., Li, L., Solomon, J., and Burnaev, E. Continuous wasserstein-2 barycenter estimation without minimax optimization. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=3tFAs5E-Pe.

Kuang, M. and Tabak, E. G. Sample-based optimal transport and barycenter problems. *Communications on Pure and Applied Mathematics*, 72(8):1581–1630, 2019.

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. Cwb, 2020. URL https://github.com/lingxiaoli94/CWB.

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

Linder-Norén, E. Pytorch-gan, 2018. URL https://github.com/eriklindernoren/PyTorch-GAN.

Liu, H., Xianfeng, G., and Samaras, D. A two-step computation of the exact gan wasserstein distance. In *International Conference on Machine Learning*, pp. 3159–3168, 2018.

Luise, G., Salzo, S., Pontil, M., and Ciliberto, C. Sinkhorn barycenters with free support via frank-wolfe algorithm. 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/9f96f36b7aae3b1ff847c26ac94c604e-Paper.pdf.

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.

Mi, L., Yu, T., Bento, J., Zhang, W., Li, B., and Wang, Y. Variational Wasserstein barycenters for geometric clustering. *arXiv preprint arXiv:2002.10543*, 2020.

Minsker, S., Srivastava, S., Lin, L., and Dunson, D. Scalable and robust bayesian inference via the median posterior. In *International conference on machine learning*, pp. 1656–1664, 2014.

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

Rabin, J., Peyré, G., Delon, J., and Bernot, M. Wasserstein barycenter and its application to texture mixing. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pp. 435–446. Springer, 2011.

Reinhard, E., Adhikhmin, M., Gooch, B., and Shirley, P. Color transfer between images. *IEEE Computer graphics and applications*, 21(5):34–41, 2001.

Rockafellar, R. T. *Convex analysis*. Number 28. Princeton university press, 1970.

Salimans, T., Zhang, H., Radford, A., and Metaxas, D. Improving GANs using optimal transport. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkQkBnJAb.

Salvatier, J., Wiecki, T. V., and Fonnesbeck, C. Probabilistic programming in python using pymc3. *PeerJ Computer Science*, 2:e55, 2016.

Seguy, V., Damodaran, B. B., Flamary, R., Courty, N., Rolet, A., and Blondel, M. Large-scale optimal transport and mapping estimation. *arXiv preprint arXiv:1711.02283*, 2017.

Solomon, J., De Goes, F., Peyré, G., Cuturi, M., Butscher, A., Nguyen, A., Du, T., and Guibas, L. Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)*, 34(4):1–11, 2015.

Srivastava, S., Cevher, V., Dinh, Q., and Dunson, D. WASP: Scalable Bayes via barycenters of subset posteriors. In *Artificial Intelligence and Statistics*, pp. 912–920, 2015.

Srivastava, S., Li, C., and Dunson, D. B. Scalable Bayes via barycenter in Wasserstein space. *The Journal of Machine Learning Research*, 19(1):312–346, 2018.

Staib, M., Claici, S., Solomon, J. M., and Jegelka, S. Parallel streaming Wasserstein barycenters. In *Advances in Neural Information Processing Systems*, pp. 2647–2658, 2017.

Taghvaei, A. and Jalali, A. 2-wasserstein approximation via restricted convex potentials with application to improved training for gans. *arXiv preprint arXiv:1902.07197*, 2019.

Villani, C. *Topics in optimal transportation*. Number 58. American Mathematical Soc., 2003.

Xie, Y., Chen, M., Jiang, H., Zhao, T., and Zha, H. On scalable and efficient computation of large scale optimal transport. *arXiv preprint arXiv:1905.00158*, 2019.

Yang, H. and Tabak, E. G. Clustering, factor discovery and optimal transport. *arXiv preprint arXiv:1902.10288*, 2019.

## A. Proof of Proposition 1

*Proof.* For fixed $\nu$ that is absolutely continuous with respect to the Lebesgue measure, and $f_i, i = 1, \ldots, N$, the solution to the inner-loop minimization problems over $g_i$ are clearly $g_i^\star = f_i^*, i = 1, \ldots, N$. The problem (8) then becomes

$$\min_{\nu} \sum_{i=1}^{N} a_i \left\{ \sup_{f_i \in \mathbf{CVX}} \{-\mathbb{E}_\nu[f_i(X)] - \mathbb{E}_{\mu_i}[f_i^*(Y)]\} + C_{\nu,\mu_i} \right\}.$$

In view of (2), it boils down to

$$\min_{\nu} \sum_{i=1}^{N} a_i W_2^2(\nu, \mu_i),$$

which is exactly the Wasserstein barycenter problem (3). Since all the marginal distributions $\mu_i$ are absolutely continuous with respect to the Lebesgue measure, their barycenter exists and is unique. This completes the proof. □

## B. Neural Wasserstein Barycenter-F

We consider a more challenging Wasserstein barycenter problem with free weights. More specifically, given a set of marginal distribution $\mu_i, i = 1, \ldots, N$, we aim to compute their Wasserstein barycenter for all the possible weights. Of course, we can utilize Algorithm 1 to solve fixed weight Wasserstein barycenter problem (9) for different weight $a$ separately. However, this will be extremely expensive if the number of weights is large. It turns out that Algorithm 1 can be adapted to obtain the barycenters for all weights in one shot. To this end, we include the weight $a$ as an input to all the neural networks $f_i, g_i$ and $h$, rendering maps $h(z, a; \theta_h), f_i(x, a; \theta_{f_i}), g_i(y, a; \theta_{g_i})$. For each fixed weight $a$, the networks $f_i, g_i$ and $h$ with this $a$ as an input solves the Barycenter problem with this weight. Apparently, $f_i, g_i$ are only required to be convex with respect to samples, not the weight $a$. Therefore, we use PICNN (Amos et al., 2017, Section 3.2) instead of FICNN for as network architectures. PICNN is an extension of FICNN that is capable of modeling functions that are convex with respect to parts of the variable.The architecture of PICNN is depicted in Figure 11. It is a $L$-layer architecture with inputs $(x, y)$. Under some proper assumptions on the weights (the feed-forward weights $\{W_l^{(z)}\}$ for $z$ are non-negative) and activation functions of the network, the map $(x, y) \to f(x, y; \theta) := z_L$ is convex over $x$. We refer the reader to (Amos et al., 2017) for more details. The problem then becomes

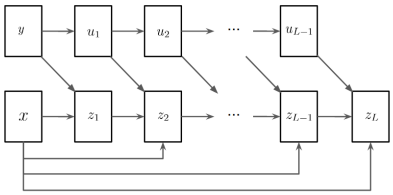



Figure 11: Partially ICNN structure

$$\min_{h} \sup_{f_i \in \mathbf{PICNN}} \inf_{g_i \in \mathbf{PICNN}} \mathbb{E}_U\{-\mathbb{E}_\eta[f_i(h(Z,a),a)] - \mathbb{E}_{\mu_i}[\langle Y, \nabla g_i(Y,a)\rangle - f_i(\nabla g_i(Y,a),a)] + \frac{1}{2}\mathbb{E}_\eta[\|h(Z,a)\|^2]\} \quad (12)$$

where $U$ is a probability distribution on the probability simplex, from which the weight $a$ is sampled. In our experiment, we used uniform distribution, but it can be any distribution that is simple to sample from, e.g., Dirichlet distribution. Effectively, the objective function in (12) amounts to the total Wasserstein cost over all the possible weights. Our formulation makes it ideal to implement stochastic gradient descent/ascent algorithm and solve the problem jointly in one training. As in the fixed weights setting, the (partial) convexity constraints of $\{g_i\}$ can be replaced by a penalty term. For batch implementation, in each batch, we randomly choose one $a \in U$ and $M$ samples $\{Y_j^i\}$ from $\mu_j$ and $\{Z_j\}$ from $\eta$. The unbiased batch estimation of the objective in (12) reads

$$\sum_{i=1}^{N} a_i\{J(\theta_{f_i}, \theta_{g_i}, \theta_h) + R(\theta_{g_i})\} + \frac{1}{2M}\sum_{j=1}^{M} ||h(Z_j, a)||^2, \quad (13)$$

where

$$J = \frac{1}{M}\sum_{j=1}^{M}[f_i(\nabla g_i(Y_j^i, a), a) - \langle Y_j^i, \nabla g_i(Y_j^i, a)\rangle - f_i(h(Z_j, a), a)],$$

and $R(\theta_{g_i}) = \lambda \sum_{W_l^{(z)} \in \theta_{g_i}} \left\| \max\left(-W_l^{(z)}, 0\right) \right\|_F^2$. By alternatively updating $h, f_i, g_i$ we establish Neural Wasserstein Barycenter-F (NWB-F) (Algorithm 2).

**Algorithm 2** Neural Wasserstein Barycenter-F

**Input** Marginal dist. $\mu_{1:N}$, Generator dist. $\eta$, Batch size $M$, weight dist. $U$
**for** $k_3 = 1, \ldots, K_3$ **do**
  Sample $a \sim U$
  Sample batch $\{Z_j\}_{j=1}^M \sim \eta$
  Sample batch $\{Y_j^i\}_{j=1}^M \sim \mu_i$
  **for** $k_2 = 1, \ldots, K_2$ **do**
    **for** $k_1 = 1, \ldots, K_1$ **do**
      Update all $\theta_{g_i}$ to decrease (13)
    **end for**
    Update all $\theta_{f_i}$ to increase (13)
    Clip: $W_l^{(z)} = \max(W_l^{(z)}, 0)$ for all $\theta_{f_i}$
  **end for**
  Update $\theta_h$ to decrease (13)
**end for**

The block diagram for Neural Wasserstein Barycenter-F (Algorithm 2) is shown in Figure 12.

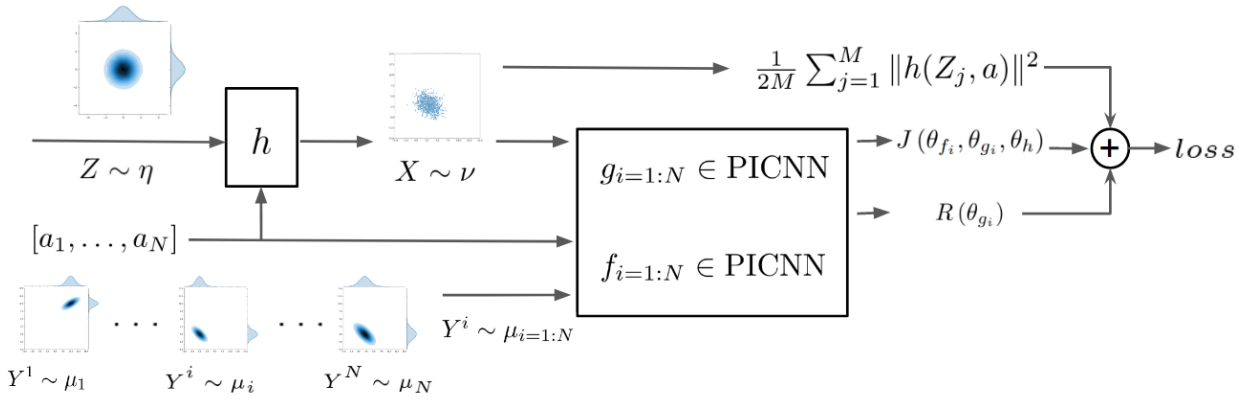
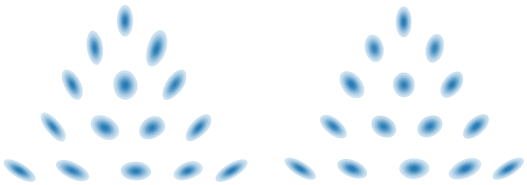



Figure 12: Block diagram for Neural Wasserstein Barycenter-F Algorithm

### B.1. Supportive experiments for NWB-F

In this part, we evaluate the performance of NWB-F which is an algorithm to calculate the Wasserstein barycenter of a given set of marginals for all weights in one shot. Departing from NWB, the networks $f_i$ and $g_i$ are of PICNN structure. We carry out 3 sets of experiments when the marginal distributions are Gaussian, Gaussian mixtures and sharp distributions. In these experiments, NWB-F converges after 15000 outer cycle iterations.

**Gaussian marginal** We present the experimental result of implementing NWB-F (Algorithm 2) to compute the Wasserstein barycenter for all combinations of weights with a single training. The result for the case of Gaussian marginal distributions, and 12 combination of weight values, is depicted in Figure 13. For comparison, we have included the exact barycenter. It is qualitatively observed that our approach is able to



(a) Ours NWB-F (b) Ground truth

Figure 13: Barycenter with different weights using NWB-F.

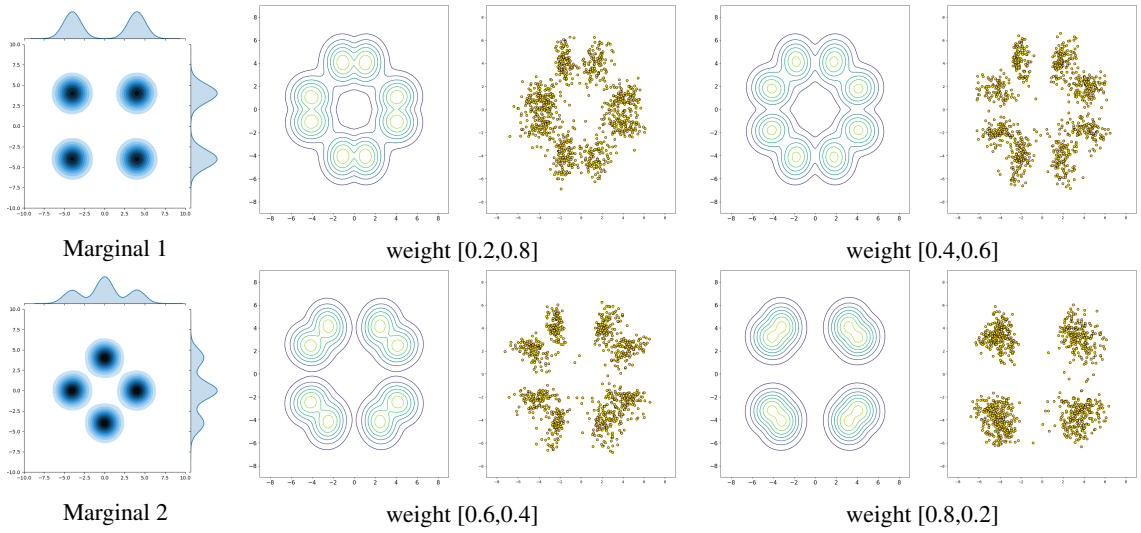



Figure 14: Barycenter with different weights using NWB-F. For each subfigure, the plot on the left is obtained using Solomon et al. (2015) and the plot on the right is obtained using NWB-F.

compute the Wasserstein barycenter for the selected weight combinations in comparison to exact barycenter. The three marginals are

$$\mu_1{=}N(\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}0.5 & 0\\0 & 2\end{bmatrix}),\ \ \mu_2{=}N(\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}2 & 1\\1 & 1\end{bmatrix}),\ \ \mu_3 = N(\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}2 & -1\\-1 & 1\end{bmatrix}).$$

To quantitatively verify the performance of NWB-F, we compare the barycenters to ground truth with several different weight in terms of KL-divergence. The resulting error is respectively $0.0235$ for $a = [0.5, 0.25, 0.25]$, $0.0153$ for $a = [0.25, 0.5, 0.25]$, and $0.0114$ for $a = [0.25, 0.25, 0.5]$. The error of results using NWB-F is consistently small among different weight combinations.

The networks $f_i$ and $g_i$ each has 3 layers and the generative network $h$ has 4 layers. All networks have 12 neurons for each hidden layer. Learning rate is 0.001. The inner loop iteration numbers are $K_1 = 6$ and $K_2 = 4$. The batch size is $M = 100$.

**Gaussian mixture marginal** We apply NWB-F to obtain the Wasserstein barycenter for all weights in one shot. The first marginal is a uniform combination of the Gaussian distributions

$$N(\begin{bmatrix}4\\4\end{bmatrix},\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}),\ \ N(\begin{bmatrix}4\\-4\end{bmatrix},\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}),\ \ N(\begin{bmatrix}-4\\-4\end{bmatrix},\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}),\ \ N(\begin{bmatrix}-4\\4\end{bmatrix},\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}).$$

The second marginal is a uniform combination of the Gaussian distributions

$$N(\begin{bmatrix}0\\4\end{bmatrix},\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}),\ \ N(\begin{bmatrix}4\\0\end{bmatrix},\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}),\ \ N(\begin{bmatrix}0\\-4\end{bmatrix},\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}),\ \ N(\begin{bmatrix}-4\\0\end{bmatrix},\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}).$$

The experiment results are depicted in Figure 14 in comparison with Convolutional Wasserstein Barycenter (Solomon et al., 2015). We remark that this is not a fair comparison since NWB-F obtained all the barycenters with different weights in one shot while Solomon et al. (2015) has to be run separately for each weight. Nevertheless, NWB-F generates reasonable results.

The networks $f_i$ and $g_i$ each has 5 layers and the generative network $h$ has 6 layers. All networks have 12 neurons for each hidden layer. Batch normalization is used in $h$. Learning rate is 0.001. The inner loop iteration numbers are $K_1 = 10$ and $K_2 = 6$. The batch size is $M = 100$.

**Sharp line marginal** Given two marginals supported on two lines, we apply NWB-F to obtain the Wasserstein barycenter for all weights in one shot. Note that these Wasserstein barycenters in fact constitute the Wasserstein geodesic between the two distributions. The networks $f_i$ and $g_i$ each has 4 layers and the generative network $h$ has 4 layers. All networks have 12

neurons for each hidden layer. Batch normalization is used in $h$. Learning rate is 0.001. The inner loop iteration numbers are $K_1 = 6$ and $K_2 = 4$. The batch size is $M = 100$. The experiment results are depicted in Figure 15 in comparison with ground truth results.

(a) Ours NWB-F

(b) Ground truth

Figure 15: Barycenter with different weights using NWB-F. For each subfigure, the two plots at both ends are given marginal distributions supported on two line segments.

## C. Experiment details for NWB and more supportive experiments

In this section, we provide the experiment details as well as more supportive experimental results of NWB. Some common experiment setup for NWB is:
1) All $f_i$ and $g_i$ networks use CELU activation function while the $h$ network uses **PReLU** (He et al., 2015) activation function.
2) The weight $\lambda = 0.1$ for the regularizer $R(\theta_{g_i}) = \lambda \sum_{W_l \in \theta_{g_i}} \|\max(-W_l, 0)\|_F^2$.
3) All optimizers are Adam.
4) All $h$ used in this article are vanilla feedforward networks.
5) All input Gaussian distribution $\eta$ has zero mean an identity covariance.
6) The inner loop iteration numbers are $K_1 = 6$ and $K_2 = 4$.
7) The batch size is $M = 100$ unless further specified.
8) NWB converges after **15000** outer cycle iterations unless further specified.

We also note that the value of the evalutation metric $\text{BW}_2^2$–UVP is sensitive to the number of samples. To be consistent, we draw 10000 samples from each method to calculate $\text{BW}_2^2$–UVP.

### C.1. Learning the Gaussian mixture Wasserstein Barycenter

In Figure 16, we further test NWB with 3 marginals of Gaussian mixtures. The first marginal is a uniform combination of 4 Gaussian components

$$N(\begin{bmatrix}4\\4\end{bmatrix},\begin{bmatrix}1&0\\0&1\end{bmatrix}),\ N(\begin{bmatrix}4\\-4\end{bmatrix},\begin{bmatrix}1&0\\0&1\end{bmatrix}),\ N(\begin{bmatrix}-4\\-4\end{bmatrix},\begin{bmatrix}1&0\\0&1\end{bmatrix}),\ N(\begin{bmatrix}-4\\4\end{bmatrix},\begin{bmatrix}1&0\\0&1\end{bmatrix}).$$

The second marginal is a uniform combination of 3 Gaussian components

$$N(\begin{bmatrix}4\\4\end{bmatrix},\begin{bmatrix}1&0\\0&1\end{bmatrix}),\ N(\begin{bmatrix}-4\\4\end{bmatrix},\begin{bmatrix}1&0\\0&1\end{bmatrix}),\ N(\begin{bmatrix}0\\-4\end{bmatrix},\begin{bmatrix}1&0\\0&1\end{bmatrix}).$$

The third marginal is a uniform combination of 3 Gaussian components

$$N(\begin{bmatrix}4\\-4\end{bmatrix},\begin{bmatrix}1&0\\0&1\end{bmatrix}),\ N(\begin{bmatrix}-4\\-4\end{bmatrix},\begin{bmatrix}1&0\\0&1\end{bmatrix}),\ N(\begin{bmatrix}0\\4\end{bmatrix},\begin{bmatrix}1&0\\0&1\end{bmatrix}).$$

For NWB, all the networks have 10 neurons for each hidden layer. The networks $f_i$ and $g_i$ each has 4 layers and the generative network $h$ has 6 layers. The initial learning rate is 0.001 and the learning rate drops 90 percent every 20 epochs. For Solomon et al. (2015), the regularization intensity is set to 0.004.

We draw 1000 samples for each scatter plot. It can be seen that NWB can better capture the different modes of the distributions.

### C.2. Learning barycenters with sharp marginal distributions

We illustrate the performance of NWB in learning the Wasserstein barycenter when the marginal distributions are sharp. The common setup is all networks have 6 neurons for each hidden layer and the input Gaussian $\eta$ dimension is 1.

**Line marginals** We follow the examples reported in Claici et al. (2018, Figure 4), where the marginal distributions are uniform distributions on 10 random two-dimensional lines as shown in Figure 17. It is observed that our algorithm is able to learn the sharp barycenter. The network $f_i$ and $g_i$ each has 4 layers and $h$ has 4 layers. $h$ network is linear. Learning rate is 0.0001. The inner loop iteration numbers are $K_1 = 6$ and $K_2 = 4$.

**Ellipse marginals** We also tested NWB on another example (Claici et al., 2018, Figure 6) to learn the barycenter of 10 uniform marginals supported on ellipses and obtained excellent results. The network $f_i$ and $g_i$ each has 5 layers and $h$ has 4 layers. The initial learning rate is 0.001 and the learning rate drops 90 percent every 15 epochs. The inner loop iteration numbers are $K_1 = 10$ and $K_2 = 6$.

### C.3. Learning the 2D and 3D Wasserstein Barycenter

This is for the results in Figure 4.

The network $f_i$ and $g_i$ each has 4 layers and $h$ has 5 layers. In $h$ network, there is a batch normalization layer before each hidden layer. All networks have 16 neurons for each hidden layer. the input Gaussian $\eta$ dimension is equal to the marginal distribution dimension.

**Circle-square example** Learning rate is 0.001.

**Block example** Learning rate is 0.001.

**Digit 3 example** $f$ and $g$ learning rate is 0.0001, and $h$ is 0.001. Learning rate drops 90 percent every outer cycle 12000 iterations. Our algorithm converges after 25000 outer cycle iterations.

### C.4. Scalability with the dimension

**Gaussian** The results are displayed in Figure 5.

The network $f_i$ and $g_i$ each has 4 layers and $h$ has 5 layers. In $h$ network, there is a batch normalization layer before each hidden layer. All networks have $\max(10, 2D)$ neurons for each hidden layer, where $D$ is the dimension of marginal distributions. The input Gaussian $\eta$ dimension is equal to the marginal distribution dimension. Learning rate is 0.001.

**MNIST 0 and 1** The results are displayed in Figure 6 and Figure 7.

The network $f_i$ and $g_i$ each has 5 layers and $h$ has 5 layers. In $h$ network, we use batch normalization and dropout (probability 0.2 to be zeroed) operation before each hidden layer. All networks have 1024 neurons for each hidden layer. The input Gaussian $\eta$ dimension is 16. Learning rate is initially 0.0001 for network $f_i$ and $g_i$; 0.001 for $h$, and drops 90 percent every 1500 outer cycle iterations. Our algorithm converges after 7500 outer cycle iterations.

**MNIST 0-4 and 5-9** To further evaluate our algorithm as a generative model for marginal distributions, we tested our algorithm on a upgraded task based on MNIST 0 and 1 experiment above. The results are shown in Figure 18. The first marginal $\mu_1$ is an empirical distribution consisting of digit 0,1,2,3,4 samples and the second marginal $\mu_2$ is for digit 5,6,7,8,9. We generate fresh samples from the barycenter using the generator $h(Z)$, where $Z \sim \mathcal{N}(\mathbf{0}, I)$. We push-forward the samples $h(Z)$ through the maps $\nabla f_1(h(Z))$ and $\nabla f_2(h(Z))$ to generate new samples from the marginal distributions. It's

expected that the panel (c) contains of only digits 0-4, and panel (d) only digits 5-9 and the results are consistent with the expectation.

The network $f_i$ and $g_i$ each has 5 layers and $h$ has 5 layers. In $h$ network, we use batch normalization before each hidden layer. All networks have 1024 neurons for each hidden layer. The input Gaussian $\eta$ dimension is 8. Learning rate is initially 0.0001 for network $f_i$ and $g_i$; 0.001 for $h$, and drops 90 percent every 25000 outer cycle iterations. The number of outer cycle iterations is set to be $100,000$.

**MNIST and USPS** We tested our algorithm NWB on different datasets: MNIST and USPS. The results are displayed in Figure 19. We resize the MNIST samples to be $16 \times 16$ to be consistent with the USPS dataset. The dimension of this problem is thus 256. MNIST shows slimmer and smaller fonts compared to USPS digits. The barycenter fuses the two dataset styles, whereas $\nabla g_i \sharp(\mu_i)$ exhibits tidier results. Figure 19 (e)-(f) show that our algorithm is able to generate new samples from both marginals (MNIST and USPS) with random Gaussian input using the same approach as in the previous example.

The network $f_i$ and $g_i$ each has 5 layers and $h$ has 6 layers. In $h$ network, we use batch normalization before each hidden layer. All networks have 512 neurons for each hidden layer. The input Gaussian $\eta$ dimension is 128. Learning rate is initially 0.0001 and drops 90 percent every 6000 outer cycle iterations. The number of outer cycle iterations is set to be 75000.

### C.5. Subset posterior aggregation

The results are displayed in Table 1. We preprocess the training data as follows (Korotin et al., 2021b): i) apply the stochastic approximation trick to each $\mu_i$ (Minsker et al., 2014); ii) remove the mean of each marginal by shifting $\tilde{Y}_i = Y_i - m(\mu_i)$, where $m(\mu_i)$ is the mean of distribution $\mu_i$ (Álvarez-Esteban et al., 2016); iii) scale each marginal distributions to be in a proper magnitude. Note that scaling won't affect $\text{BW}_2^2$–UVP value. We use the same data preprocessing methods for other barycenter methods. The network $f_i$ and $g_i$ each has 5 layers and $h$ has 5 layers. In $h$ network, there is a batch normalization layer before each hidden layer. All networks have 10 neurons for each hidden layer. The input Gaussian $\eta$ dimension is 8. Learning rate is 0.01. Our algorithm converges after 8000 outer cycle iterations.

### C.6. Color palette averaging

The results are displayed in Figure 8 and Figure 9. The batch size is $M = 1200$. The network $f_i$ and $g_i$ each has 4 layers and $h$ has 5 layers. In $h$ network, there is a batch normalization layer before each hidden layer. $f_i$ and $g_i$ networks have 16 neurons for each hidden layer, and $h$ has 32 neurons for each hidden layer. The input Gaussian $\eta$ dimension is 3. Learning rate is 0.001. Our algorithm converges after 100000 outer cycle iterations.

### C.7. Serving as a Generative Adversarial Model in the one marginal setting

WGAN and WGAN-GP results are generated using Pytorch-GAN library (Linder-Norén, 2018). W2GN results are generated using Korotin (2020) and adopt DenseICNN architecture proposed in the paper. We refer the reader to the Korotin et al. (2021a, Section B.2, Section C.1) for DenseICNN and pretrain details. The number of total training samples for both two experiments is 60000.

**Gaussian mixture** The results are displayed in Figure 3.

For NWB, the networks $f_i$ and $g_i$ each has 5 layers and the generative network $h$ has 6 layers. All networks have 10 neurons for each hidden layer. The initial learning rate is 0.001. The batch size is $M = 60$.

For WGAN and WGAN-GP, they all use fully-connected linear layers and ReLU activation function. All discriminators and generators have 4 layers and 512 neurons for each hidden layer. Learning rate is 0.0001. The batch size is 256. The number of total iteration is 50000.

For W2GN, all netorks use DenseICNN $[3; 128, 128, 64]$ architecture. Here 3 is the rank of each input-quadratic skip-connection's Hessian matrix. Each following number represents the size of a hidden dense layer in the sequential part of the network. The batch size is 1024. Learning rate is initially 0.001 and drops 90 percent every 25000 iterations. The number of total iteration is 50000.

**MNIST** The results are displayed in Figure 10. We normalize MNIST pixel values to be in range $[-1, 1]$ before training.

For NWB, the network $f_i$ and $g_i$ each has 5 layers and $h$ has 6 layers. In $h$ network, there is a batch normalization layer before each hidden layer. All networks have 1024 neurons for each hidden layer. The input Gaussian $\eta$ dimension is 64. Learning rate is initially 0.0001 and drops 90 percent every 100 epochs. The total epoch is set to be 500 epochs.

For WGAN and WGAN-GP, to be fair, they all use the same batch size, batch-normalization and fully-connected linear layers as NWB. The activation function is LeakyReLU. From input layer to output layer, the generator neuron for each layer is $[100, 128, 256, 512, 1024, 784]$; and the discriminator is $[784, 512, 256, 1]$. The final layer for the generator is also tanh. Learning rate is initially 0.0001 and drops 90 percent every 300 epochs. The total epoch is set to be 1500 epochs.

For W2GN, all networks use DenseICNN $[2; 2048, 2048, 2048]$ architecture. The batch size is also 100. Learning rate is initially 0.0001 and drops 90 percent every 300 epochs. The total epoch is set to be 1500 epochs.

## D. Experiment details for CDWB (Cuturi & Doucet, 2014, Section 4.4)

We use POT library (Flamary & Courty, 2017) and adopt Earth Movers distance solver (Bonneel et al., 2011) when solving OT programming in the inner loop.

## E. Experiment details for CRWB (Li et al., 2020)

We use the code given by Li (2020). We use quadratic regularization, which is empirically more stable than entropic regularization. We set potential networks as fully connected neural networks. The hidden layer sizes are given by

$$[\max(128, 2D), \max(128, 2D), \max(128, 2D)],$$

where $D$ is the marginal distribution dimension. The activation functions are all ReLU. The batch size as 1024. We use Adam optimizer with fixed learning rate $10^{-4}$ for Bayesian inference and MNIST examples and $10^{-3}$ for Gaussian examples. We use Monge map to recover barycenter samples (Li et al., 2020, Equation (13)). The total number of iterations is set to 50000.

## F. Experiment details for CWB (Korotin et al., 2021b)

We use the code in `https://openreview.net/forum?id=3tFAs5E-Pe`. As mentioned in the Korotin et al. (2021b, Section A), we also pretrain the potential networks as an initialization step. We use Adam optimizer with fixed learning rate $10^{-4}$ for Bayesian inference and MNIST examples and $10^{-3}$ for Gaussian examples. Other setups are exactly the same as the (Korotin et al., 2021b, Section C.4.1).

## G. Optimization landscape for class of quadratic functions

In order to understand and compare various optimization formulations to estimate Wasserstein barycenter, it is insightful to examine them for special cases where analysis is feasible. For this purpose, we study the optimization landscape of our formulation and (Korotin et al., 2021b) in the special case where the class of functions is restricted to quadratic functions. In particular, we show that our optimization formulation simplifies to a smooth concave-convex-concave optimization for this special case, while the formulation of (Korotin et al., 2021b) is non-smooth and non-convex.

We consider the simplest case that the functions are parameterized as follows: $h(z) = z + \alpha$, $f_i(x) = \frac{1}{2}\|x\|^2 + \beta_i^T x$, and $g_i(y) = \frac{1}{2}\|y\|^2 + \gamma_i^T y$, where $\alpha, \beta_i, \gamma_i \in \mathbb{R}^n$ are the parameters that serve as optimization variables. Then, in this case, the optimization problem (9) simplifies to

$$\min_{\alpha} \max_{\{\beta_i\}_{i=1}^N} \min_{\{\gamma_i\}_{i=1}^N} \sum_{i=1}^N a_i \left[\frac{1}{2}\|\gamma_i\|^2 + \beta_i^T(\gamma_i + m_i - \alpha)\right] \tag{14}$$

where $m_i = \mathbb{E}_{\mu_i}[Y^i]$. The objective function is convex in $\gamma_i$, linear in $\beta_i$ and $\gamma_i$. Inserting the optimal value for $\gamma_i = -\beta_i$ yields

$$\min_{\alpha} \max_{\{\beta_i\}_{i=1}^N} \sum_{i=1}^N a_i \left[-\frac{1}{2}\|\beta_i\|^2 + \beta_i^T(m_i - \alpha)\right]. \tag{15}$$

This is concave in $\beta_i$. Inserting the optimal value $\beta_i = m_i - \alpha$ yields

$$\min_{\alpha} \frac{1}{2}\|\alpha\|^2 - \alpha^T \sum_{i=1}^{N} a_i m_i + \sum_{i=1}^{N} a_i \|m_i\|^2 \tag{16}$$

which is convex in $\alpha$ with optimal value at $\alpha = \sum_{i=1}^{N} a_i m_i$. This means that the optimal generator $h(z) = z + \sum_{i=1}^{N} a_i m_i$ learns average mean of the marginal distributions. This is the exact Wassserstein barycenter for the case that marginal distributions are Gaussian distributions with the same covariance.

In contrast, consider the optimization formulation of (Korotin et al., 2021b, Eq. 14) with the following parameterization: $\psi_i^{\dagger}(x) = \frac{1}{2}\|x\|^2 + \alpha_i^T x$ and $\bar{\psi}_i^{\dagger\dagger}(x) = \frac{1}{2}\|x\|^2 + \beta_i^T x$. Then, the optimization problem simplifies to

$$\min_{\{\alpha_i\}_{i=1}^N, \{\beta_i\}_{i=1}^N} \sum_{i=1}^{N} a_i \left[-\frac{1}{2}\|\alpha_i\|^2 - \alpha_i^T \beta_i - m_i^T \beta_i\right] + \tau \mathbb{E}_{\hat{P}}\left[\left(\sum_{i=1}^{N} a_i \beta_i^T Y\right)_+\right] + \lambda \sum_{i=1}^{N} a_i \|\alpha_i + \beta_i\|^2 \tag{17}$$

where $\hat{P}$ is a distribution that should be chosen such that $\tau\hat{P}$ is larger than the barycenter density (with $\tau > 1$). Although the optimization problem involves single minimization compared to our min-max-min formulation, the optimization objective is much more complicated. Our first observation is that the optimization problem is not convex in $\alpha_i$ if $\lambda < \frac{1}{2}$ (the optimization algorithm diverges). Inserting the optimal value $\alpha_i = -\beta_i$, the optimization becomes

$$\min_{\{\beta_i\}_{i=1}^N} \sum_{i=1}^{N} a_i \left[\frac{1}{2}\|\beta_i\|^2 - m_i^T \beta_i\right] + \tau \mathbb{E}_{\hat{P}}\left[\left(\sum_{i=1}^{N} a_i \beta_i^T Y\right)_+\right]. \tag{18}$$

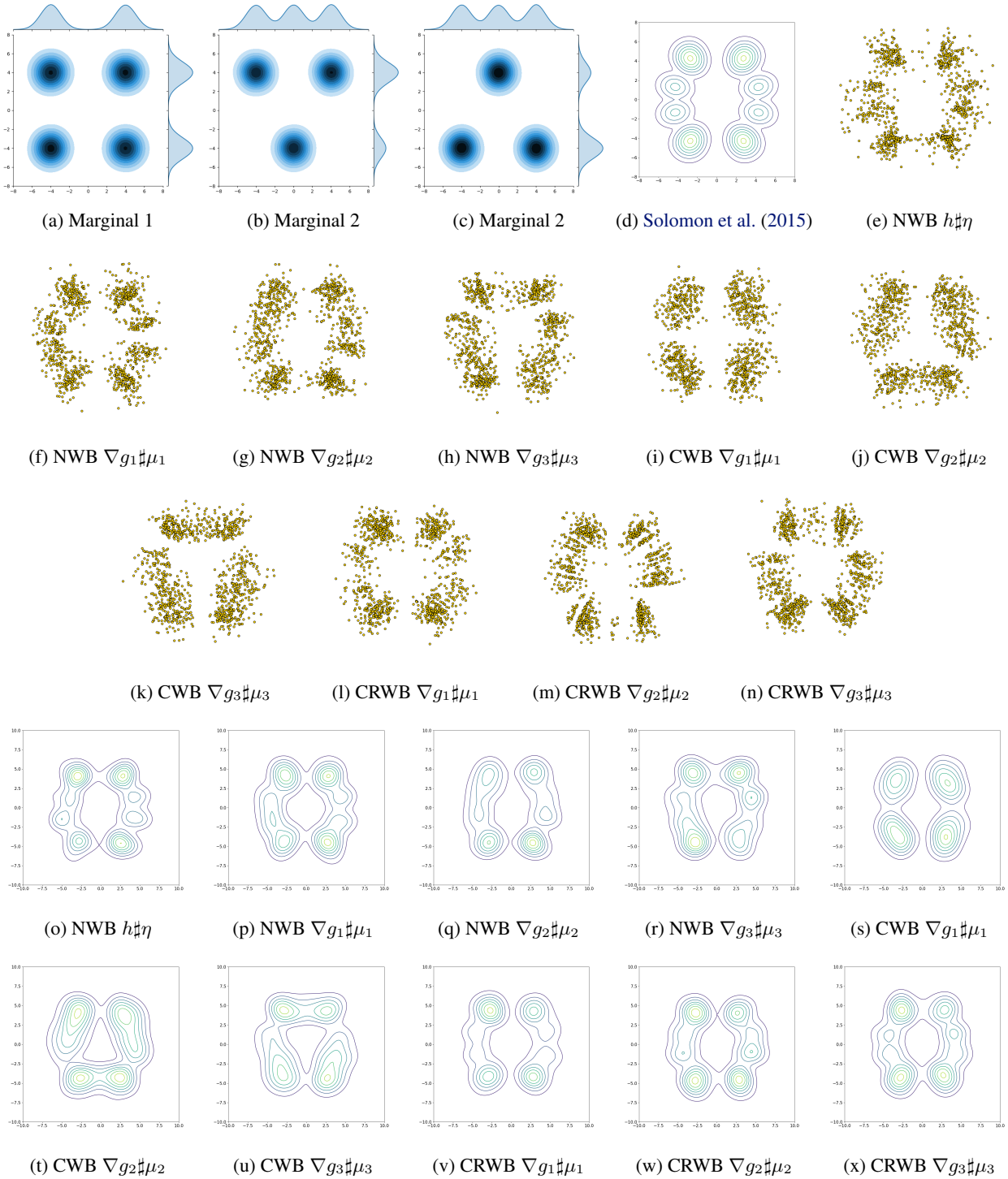
This is convex, but non-smooth optimization problem in $\beta_i$. In order for the expected solution $\beta_i = m_i - \sum_{i=1}^{N} a_i m_i$ be optimal for this problem, it must satisfy the first-order optimality condition

$$a_i(\beta_i - m_i) + \tau a_i \partial l(0) = 0$$

where $l(\xi) := \mathbb{E}_{\hat{P}}\left[(\xi Y)_+\right]$ and $\partial l(0)$ denotes an element in sub-differential of $l(\xi)$ at $\xi = 0$. The function $l(\xi) = \mu_+ \xi$ for $\xi > 0$ and $l(\xi) = -\mu_- \xi$ for $\xi < 0$, where $\mu_+ = \mathbb{E}_{\hat{P}}\left[(Y)_+\right]$ and $\mu_- = \mathbb{E}_{\hat{P}}\left[(-Y)_+\right]$. As a result, the sub-differential $\partial l(0) \in [-\mu_-, \mu_+]$. Therefore, summing the first-order optimality condition for $i = 1, \ldots, N$ implies

$$\sum_{i=1}^{N} a_i m_i \in [-\tau\mu_-, \tau\mu_+].$$

Although, this condition holds when $\hat{P}$ is Gaussian centered at the barycenter location $\sum_{i=1}^{N} a_i m_i$ (with $\tau > 1$), it may not hold with other $\hat{P}$. For example, if $\hat{P}$ is $N(0, 1)$, then $\mu_+ = \mu_- = \frac{1}{\sqrt{2\pi}}$, and the condition does not hold if $\sum_{i=1}^{N} a_i m_i > \frac{\tau}{\sqrt{2\pi}}$.



(a) Marginal 1 (b) Marginal 2 (c) Marginal 2 (d) Solomon et al. (2015) (e) NWB $h\sharp\eta$

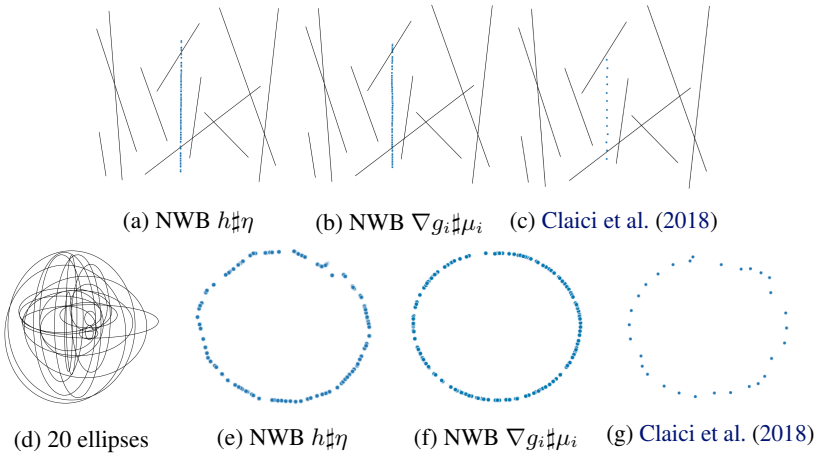
(f) NWB $\nabla g_1\sharp\mu_1$ (g) NWB $\nabla g_2\sharp\mu_2$ (h) NWB $\nabla g_3\sharp\mu_3$ (i) CWB $\nabla g_1\sharp\mu_1$ (j) CWB $\nabla g_2\sharp\mu_2$

(k) CWB $\nabla g_3\sharp\mu_3$ (l) CRWB $\nabla g_1\sharp\mu_1$ (m) CRWB $\nabla g_2\sharp\mu_2$ (n) CRWB $\nabla g_3\sharp\mu_3$

(o) NWB $h\sharp\eta$ (p) NWB $\nabla g_1\sharp\mu_1$ (q) NWB $\nabla g_2\sharp\mu_2$ (r) NWB $\nabla g_3\sharp\mu_3$ (s) CWB $\nabla g_1\sharp\mu_1$

(t) CWB $\nabla g_2\sharp\mu_2$ (u) CWB $\nabla g_3\sharp\mu_3$ (v) CRWB $\nabla g_1\sharp\mu_1$ (w) CRWB $\nabla g_2\sharp\mu_2$ (x) CRWB $\nabla g_3\sharp\mu_3$

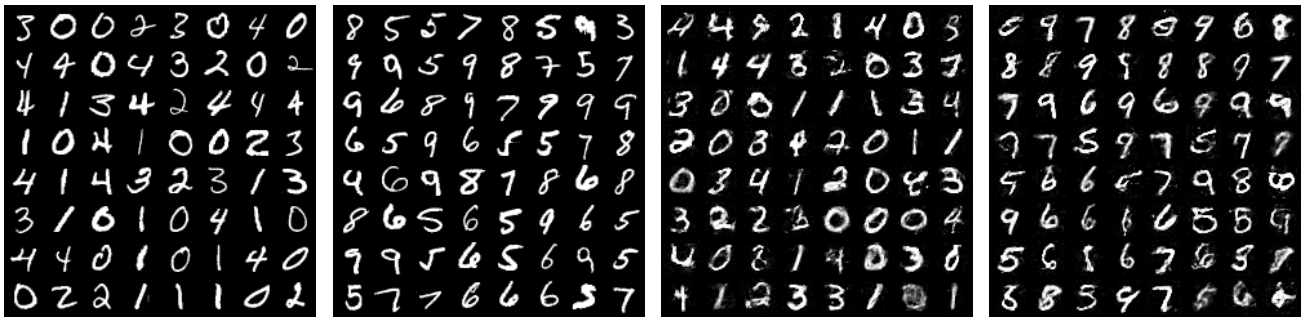
Figure 16: Wasserstein barycenter of two Gaussian mixture marginals. Both of scatter plots and the level sets are exhibited.



(a) NWB $h\sharp\eta$ (b) NWB $\nabla g_i\sharp\mu_i$ (c) Claici et al. (2018)

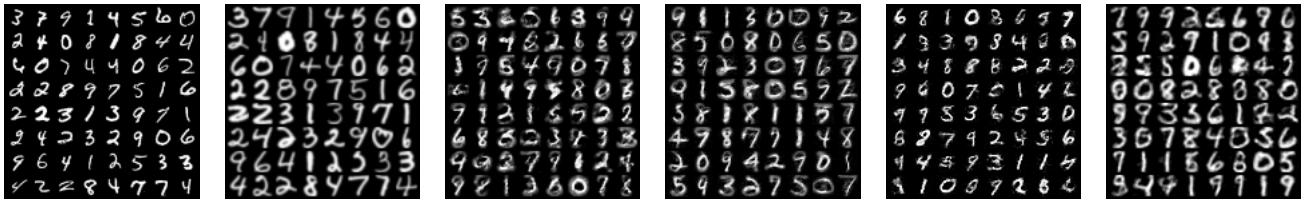
(d) 20 ellipses (e) NWB $h\sharp\eta$ (f) NWB $\nabla g_i\sharp\mu_i$ (g) Claici et al. (2018)

Figure 17: (a)-(c):Wasserstein barycenter of 10 distributions supported on random lines; (d)-(g):Wasserstein barycenter of 10 uniform marginal distributions supported on random ellipses shown in (d). 200 points are sampled from estimated barycenter. 13 points and 30 points are sampled from line and ellipse barycenter through the Claici et al. (2018) because this is the maximum number of points allowed for the Claici et al. (2018) to terminate in a reasonable amount of time.



(a) $\mu_1$: MNIST 0-4 digit (b) $\mu_2$: MNIST 5-9 digit (c) backward to $\mu_1$ (d) backward to $\mu_2$

Figure 18: MNIST 0-4 and 5-9 barycenter (784-dimensional problem): (a)-(b) Marginal distributions consisting of 0-4 and 5-9 digits; (c)-(d) Generating digit 0-4 and 5-9 from random input $Z \sim \mathcal{N}(\mathbf{0}, I)$ using our architecture with the map $\nabla f_i(h(Z))$.



(a) $\mu_1$: MNIST (b) $\mu_2$: USPS (c) NWB $h\sharp\eta$ (d) NWB $\nabla g_i\sharp(\mu_i)$ (e) backward to $\mu_1$ (f) backward to $\mu_2$

Figure 19: USPS and MNIST barycenter (256-dimensional problem): (a)-(b) Marginal distributions consisting of MNIST and USPS digits;(c)-(d) NWB generates barycenter by generator $h$ and pushforward $\nabla g_i$; (e)-(f) Generating MNIST and USPS digits from random input $Z \sim \mathcal{N}(\mathbf{0}, I)$ using our architecture with the map $\nabla f_i(h(Z))$.