A. Proof of Proposition 1

Proof. For fixed ν 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 μ_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

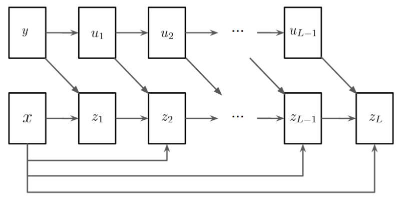


Figure 11: Partially ICNN structure

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

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 μ_j and $\{Z_j\}$ from η. 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

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

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 $k_3 = 1, \dots, K_3$ 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 $k_2 = 1, \dots, K_2$ do
        for $k_1 = 1, \dots, K_1$ 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.

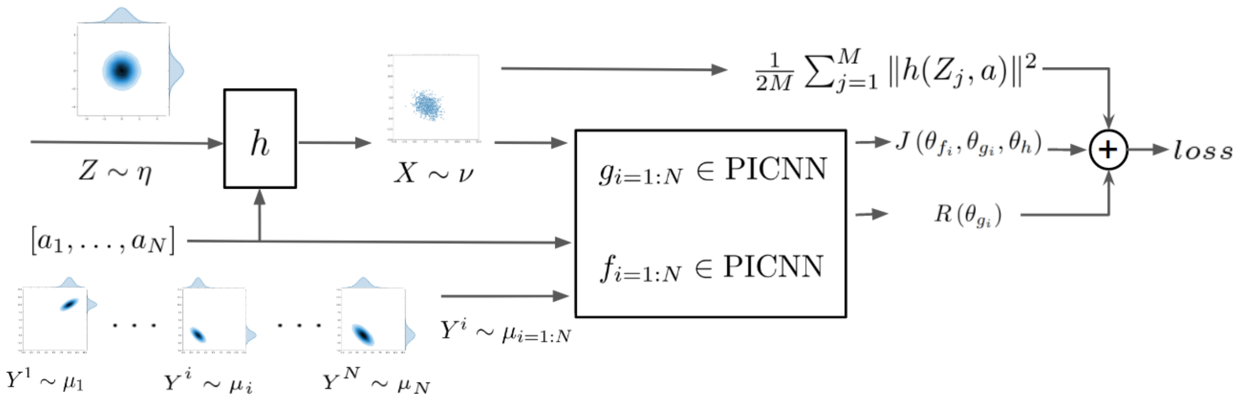
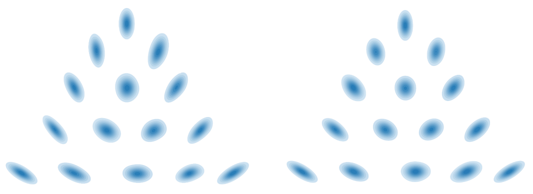


Figure 12: Block diagram for Neural Wasserstein Barycenter-F Algorithm

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.

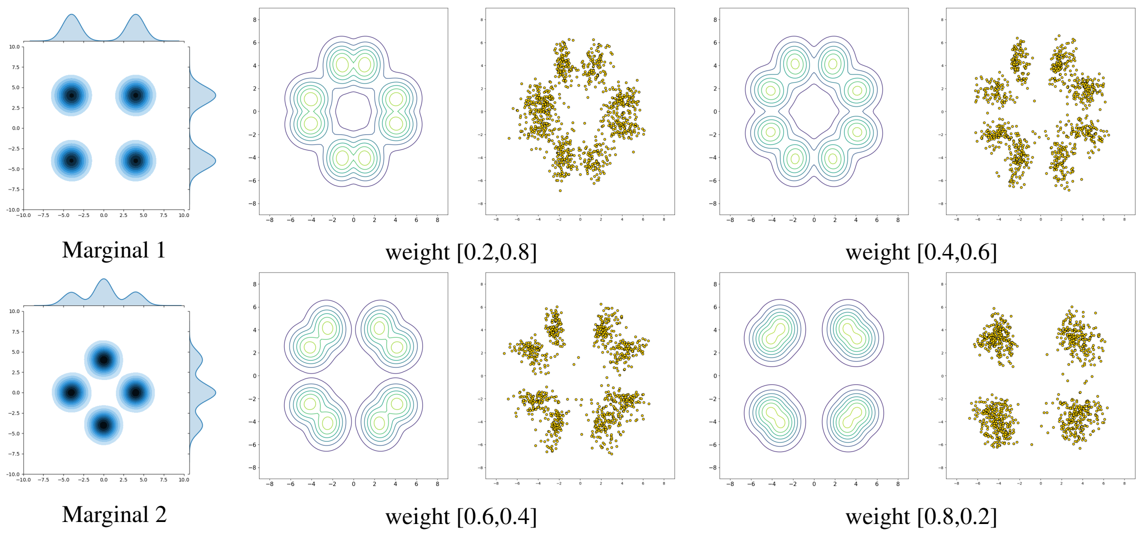


Figure 14: Barycenter with different weights using NWB-F. For each subfigure, the plot on the left is obtained using Solomon et al. (2015) and the plot on the right is obtained using NWB-F.

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

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

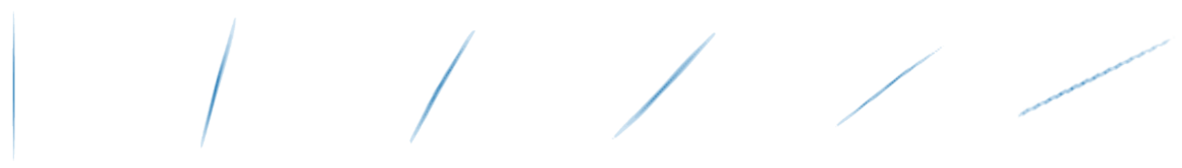
The second marginal is a uniform combination of the Gaussian distributions

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

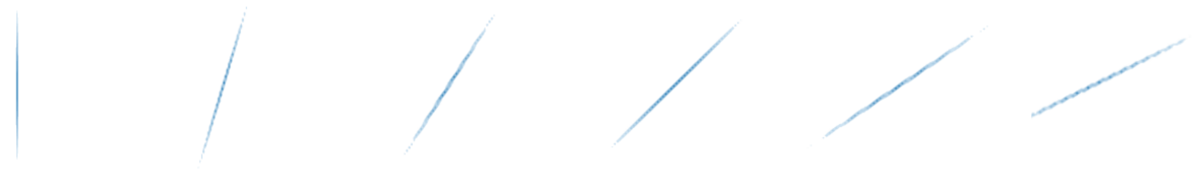
The experiment results are depicted in Figure 14 in comparison with Convolutional Wasserstein Barycenter (Solomon et al., 2015). We remark that this is not a fair comparison since NWB-F obtained all the barycenters with different weights in one shot while Solomon et al. (2015) has to be run separately 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 η 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 BW_2^2–UVP is sensitive to the number of samples. To be consistent, we draw 10000 samples from each method to calculate 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 η 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 η 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 η 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 η 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 μ_1 is an empirical distribution consisting of digit 0,1,2,3,4 samples and the second marginal μ_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 η 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 16×16 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 η 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 μ_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 μ_i (Álvarez-Esteban et al., 2016); iii) scale each marginal distributions to be in a proper magnitude. Note that scaling won't affect 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 η 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 η 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 η 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 γ_i, linear in β_i and γ_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 β_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 α 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 α_i if $\lambda < \frac{1}{2}$ (the optimization algorithm diverges). Inserting the optimal value $\alpha_i = -\beta_i$, the optimization becomes

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

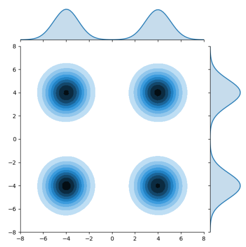
This is convex, but non-smooth optimization problem in β_i. In order for the expected solution $\beta_i = m_i - \sum_{i=1}^N a_i m_i$ be optimal for this problem, it must satisfy the first-order optimality condition

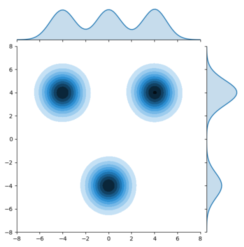
$$a_i(\beta_i - m_i) + \tau a_i \partial l(0) = 0$$

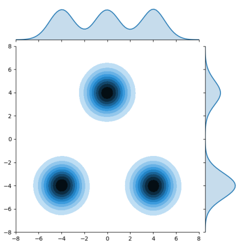
where $l(\xi) := \mathbb{E}_{\hat{P}} \left[(\xi Y)_+ \right]$ and $\partial l(0)$ denotes an element in sub-differential of $l(\xi)$ at $\xi = 0$. The function $l(\xi) = \mu_+ \xi$ for $\xi > 0$ and $l(\xi) = -\mu_- \xi$ for $\xi < 0$, where $\mu_+ = \mathbb{E}_{\hat{P}} \left[(Y)_+ \right]$ and $\mu_- = \mathbb{E}_{\hat{P}} \left[(-Y)_+ \right]$. 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

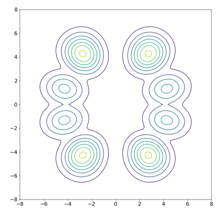
$$\sum_{i=1}^N a_i m_i \in [-\tau\mu_-, \tau\mu_+].$$

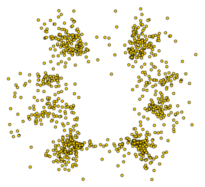
Although, this condition holds when $\hat{P}$ is Gaussian centered at the barycenter location $\sum_{i=1}^N a_i m_i$ (with $\tau > 1$), it may not hold with other $\hat{P}$. For example, if $\hat{P}$ is $N(0,1)$, then $\mu_+ = \mu_- = \frac{1}{\sqrt{2\pi}}$, and the condition does not hold if $\sum_{i=1}^N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

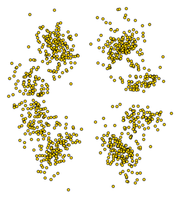

(a) Marginal 1

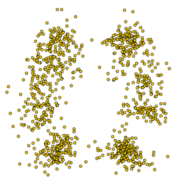

(b) Marginal 2

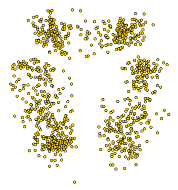

(c) Marginal 2

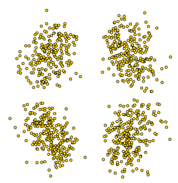

(d) Solomon et al. (2015)

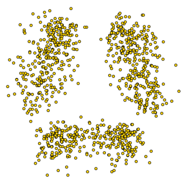

(e) NWB $h\sharp\eta$

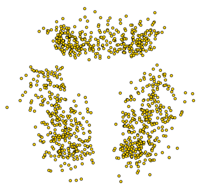

(f) NWB $\nabla g_1\sharp\mu_1$

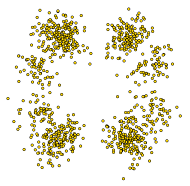

(g) NWB $\nabla g_2\sharp\mu_2$

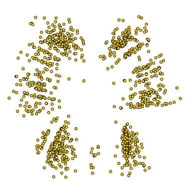

(h) NWB $\nabla g_3\sharp\mu_3$

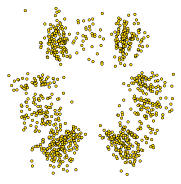

(i) CWB $\nabla g_1\sharp\mu_1$

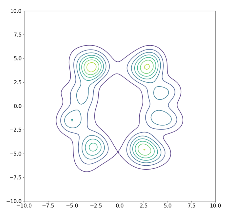

(j) CWB $\nabla g_2\sharp\mu_2$

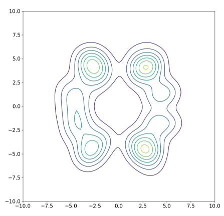

(k) CWB $\nabla g_3\sharp\mu_3$

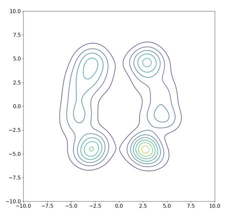

(l) CRWB $\nabla g_1\sharp\mu_1$

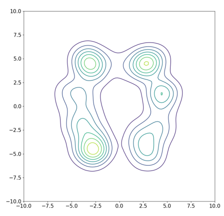

(m) CRWB $\nabla g_2\sharp\mu_2$

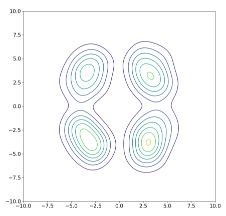

(n) CRWB $\nabla g_3\sharp\mu_3$

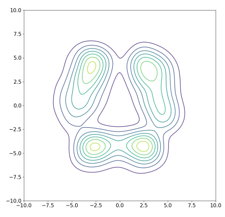

(o) NWB $h\sharp\eta$

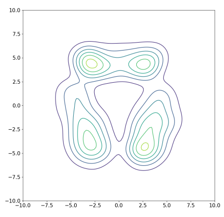

(p) NWB $\nabla g_1\sharp\mu_1$

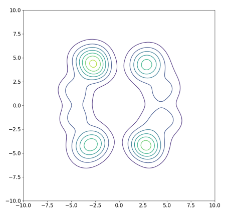

(q) NWB $\nabla g_2\sharp\mu_2$

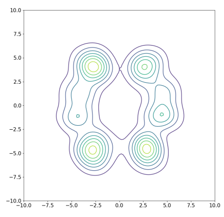

(r) NWB $\nabla g_3\sharp\mu_3$


(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 $\nabla g_2\sharp\mu_2$

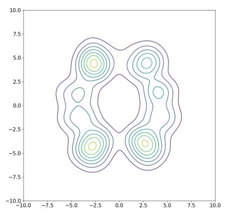
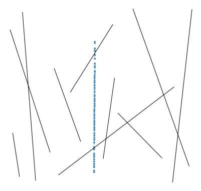
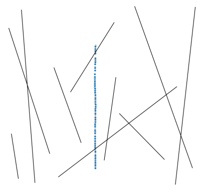
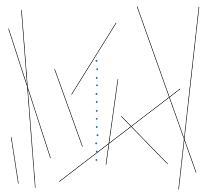
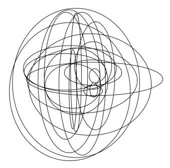

(x) CRWB $\nabla g_3\sharp\mu_3$

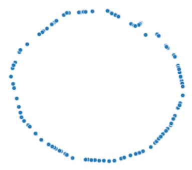
Figure 16: Wasserstein barycenter of two Gaussian mixture marginals. Both of scatter plots and the level sets are exhibited.

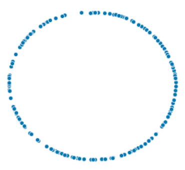

(a) NWB $h\sharp\eta$


(b) NWB $\nabla g_i\sharp\mu_i$


(c) Claici et al. (2018)


(d) 20 ellipses


(e) NWB $h\sharp\eta$


(f) NWB $\nabla g_i\sharp\mu_i$

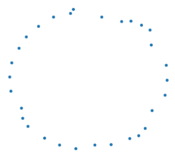
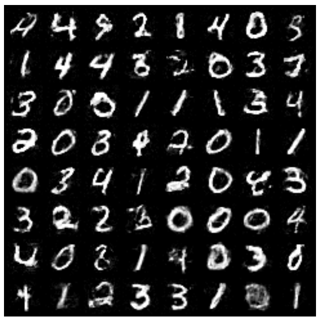

(g) Claici et al. (2018)

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) μ_1: MNIST 0-4 digit

(b) μ_2: MNIST 5-9 digit


(c) backward to μ_1

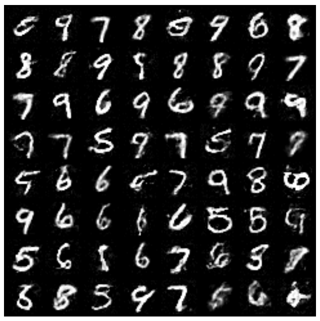
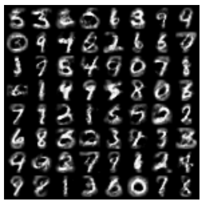
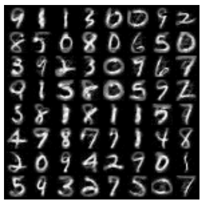

(d) backward to μ_2

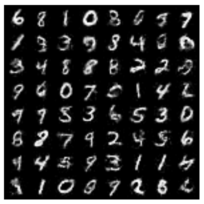
Figure 18: MNIST 0-4 and 5-9 barycenter (784-dimensional problem): (a)-(b) Marginal distributions consisting of 0-4 and 5-9 digits; (c)-(d) Generating digit 0-4 and 5-9 from random input $Z \sim \mathcal{N}(\mathbf{0}, I)$ using our architecture with the map $\nabla f_i(h(Z))$.

(a) μ_1: MNIST

(b) μ_2: USPS


(c) NWB $h\sharp\eta$


(d) NWB $\nabla g_i\sharp(\mu_i)$


(e) backward to μ_1

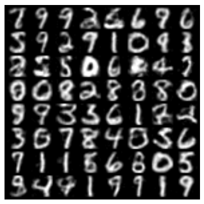

(f) backward to μ_2

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 ∇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 $\nabla f_i(h(Z))$.