Iterative Alignment Flows

Zeyu Zhou
Purdue University

Ziyu Gong
Purdue University

Pradeep Ravikumar
Carnegie Mellon University

David I. Inouye
Purdue University

Abstract

The unsupervised task of aligning two or more distributions in a shared latent space has many applications including fair representations, batch effect mitigation, and unsupervised domain adaptation. Existing flow-based approaches estimate multiple flows independently, which is equivalent to learning multiple full generative models. Other approaches require adversarial learning, which can be computationally expensive and challenging to optimize. Thus, we aim to jointly align multiple distributions while avoiding adversarial learning. Inspired by efficient alignment algorithms from optimal transport (OT) theory for univariate distributions, we develop a simple iterative method to build deep and expressive flows. Our method decouples each iteration into two subproblems: 1) form a variational approximation of a distribution divergence and 2) minimize this variational approximation via closed-form invertible alignment maps based on known OT results. Our empirical results give evidence that this iterative algorithm achieves competitive distribution alignment at low computational cost while being able to naturally handle more than two distributions.

1 INTRODUCTION

The task of aligning two or more distributions in a shared latent space without any pairing information between data points (i.e., unsupervised) has attracted increasing interest due to its varied applications. These include fair representations (Zemel et al., 2013), batch effect mitigation (Haghverdi et al., 2018), unsupervised domain adaptation (Hu et al., 2018), and generative models (Grover et al., 2020). For example, Zemel et al. (2013) estimate a shared latent representation of the class-conditional distributions that simultaneously obfuscates any information about protected attributes (e.g., race) while preserving all other information useful for classification. For genetic data, Haghverdi et al. (2018) attempt to mitigate batch effects (i.e., irrelevant shifts in the data between batches caused by non-biological factors) by estimating a shared representation among batches; this enables the integration and analysis of multiple datasets collected at different laboratories. Hu et al. (2018) perform unsupervised domain adaptation by mapping the source and target domains to a shared latent representation.

Prior work on this unsupervised alignment task generally falls into two categories: adversarial and flow-based methods. Zhu et al. (2017) propose CycleGAN for domain translation via Generative Adversarial Networks (GANs) (Goodfellow et al., 2014). Specifically, they jointly train two GANs that attempt to generate one dataset from the other dataset and add a cycle consistency loss to encourage that translating from domain A to B and back to A will yield the original point—i.e., an approximate invertibility constraint. Grover et al. (2020) propose AlignFlow that uses normalizing flows (Rezende and Mohamed, 2015; Dinh et al., 2015) to satisfy cycle consistency by construction and, in contrast to CycleGAN, learns a shared latent representation of the two datasets. AlignFlow combines both adversarial learning and maximum likelihood estimation (MLE) for training. Both CycleGAN and AlignFlow leverage adversarial learning to achieve good results. However, a fundamental limitation of adversarial learning is that it can be computationally expensive and challenging to optimize (e.g., Lucic et al., 2018; Kurach et al., 2019). To avoid adversarial learning, AlignFlow can be set to only use the MLE loss terms. In this case, the two flow models are estimated independently and they use Gaussian distribution as their latent representation which does not preserve any shared structure (e.g., black pixels in MNIST digits). Thus, without adversarial learning, a natural consequence is that AlignFlow must essentially estimate two full generative models rather

Proceedings of the 25[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

Table 1: Comparison with other methods. *AlignFlow relies on adversarial learning to get good results.

	CycleGAN	AlignFlow	LRMF	SINF	INB
Distribution alignment	✓	✓	✓	✗	✓
No adversarial learning	✗	✗*	✓	✓	✓
Iterative learning	✗	✗	✗	✓	✓
Multiple distributions	✗	✗	✗	✗	✓
Shared latent space	✗	✓	✗	✗	✓

than merely estimating the translation map—which will be simpler if the datasets share some structure. Hence, their method is likely to require higher sample complexity and computational cost. Another flow-based method LRMF (Usman et al., 2020) directly learns the transformation between two distributions via minimizing the non-adversarial log-likelihood ratio. However it is limited to the alignment between two distributions and does not learn a shared latent space.

Inspired by the limitation of existing methods, we aim at the *joint* estimation of multiple flow models that map to a shared representation *without* adversarial learning. While in general this is hard for complex datasets, simple cases can be solved very efficiently using the tools from optimal transport (OT) theory. Specifically, for 1D distributions, it is easy to compute the invertible maps between each distribution and the barycenter distribution, which naturally serves as a shared latent space. Thus, we propose a method we call Iterative Naïve Barycenter (INB), which instead of trying to solve a large global problem directly, iteratively solves simpler subproblems that first estimate a variational divergence and then minimize this variational divergence via OT barycenter maps by leveraging known efficient solutions. We leverage the development of the maximum K-Sliced Wasserstein distance proposed in Sliced Iterative Normalizing Flows (SINF) (Dai and Seljak, 2021).

As we show in the experiments, our INB method can achieve competitive or better alignment performance than baselines within a much shorter time. Moreover, INB naturally works with multiple distributions in a symmetric way which significantly reduces the computational cost and improves the alignment performance.

For clarity, we compare INB with prior methods in Table 1 and summarize our contributions as follows:

- We first develop a symmetric Monge map problem and a multi-distribution divergence to enable multi-distribution alignment.We show that the symmetric Monge map problem is equivalent to finding the Monge maps to the barycenter distribution and can be solved in closed-form for 1D distributions.
- We propose an efficient iterative algorithm for unsupervised distribution alignment by iteratively minimizing the multi-distribution divergence. Our algorithm involves two steps: the first step forms a variational approximation of the divergence around the current iterate and the second step exactly minimizes this variational divergence via known OT solutions for 1D.
- To the best of our knowledge, our INB approach is the first flow-based distribution alignment approach that can be naturally applied to align multiple distributions.
- We demonstrate the benefits of our INB approach on synthetic and real-world datasets.

2 BACKGROUND

Given samples from M class distributions $(P_{X_1}, P_{X_2}, \cdots, P_{X_M})$, our goal is to find invertible maps $T_1, T_2, \cdots, T_M$ such that the resulting latent distributions are aligned in a shared latent space, i.e., $P_{T_1(X_1)} = P_{T_2(X_2)} = \cdots = P_{T_M(X_M)}$. Because the maps are invertible, this also enables translation between any two component distributions merely by composing one map and the inverse of the other, i.e., to translate from m to m', the following map can be used $T_{m\to m'} = T_{m'}^{-1} \circ T_m$. We can formalize our alignment goal as the following optimization problem:

$$\min_{T_1,\cdots,T_M} \phi(P_{T_1(X_1)}, P_{T_2(X_2)}, \cdots, P_{T_M(X_M)}), \qquad (1)$$

where ϕ is a multi-distribution statistical divergence (i.e., a functional that is always non-negative and zero if and only if all distributions are equal). To solve this problem, we need a tractable divergence ϕ and a tractable method for optimizing this problem. We first review key concepts from optimal transport (OT) that will be needed for deriving our iterative algorithm, particularly closed-form OT solutions to 1D problems. Then, we will review tractable two-distribution divergences, which we will extend to the M distribution case in later sections.

2.1 Optimal Transport Fundamentals

We will first review some standard OT definitions. The following classical Monge map problem (Peyré and Cuturi, 2019, Remark 2.7) can be seen as finding the lowest transportation cost map that aligns the distributions (which is an explicit constraint in the problem).

Definition 1 (Monge problem)**.** *Given two distributions (P_{X_1}, P_{X_2}) supported on two spaces $(\mathcal{X}_1, \mathcal{X}_2)$ and a cost function $c(\cdot,\cdot)$, the Monge problem is defined as finding the map $T\colon \mathcal{X}_1 \to \mathcal{X}_2$ that solves:* $\arg\min_T \mathbb{E}_{P_{X_1}}[c(x, T(x))]$ s.t. $P_{T(X_1)} = P_{X_2}$, *where the objective is the* transportation cost *and the constraint is a distribution alignment condition (also known as the* pushforward condition*).*

We next review the definitions of the Kantorovich relaxation (Peyré and Cuturi, 2019, Remark 2.13) and the barycenter distribution (Peyré and Cuturi, 2019, Remark 9.1), which will be important for our development of multi-distribution divergences. For this paper, we will assume $c(x, y) = \|x - y\|_2^2$ and that one of the distributions has a density so that the barycenter is unique (Agueh and Carlier, 2011).

Definition 2 (Kantorovich Relaxation)**.** *Given the same variables as Def. 1, the Kantorovich problem is defined as:* $\mathcal{L}_c(P_{X_1}, P_{X_2}) \triangleq \min_{Q \in \mathcal{U}(P_{X_1}, P_{X_2})} \mathbb{E}_Q[c(x_1, x_2)]$, *where Q is a joint distribution over $\mathcal{X}_1$ and $\mathcal{X}_2$ such that the marginals are equal to P_{X_1} and P_{X_2} respectively (denoted by $\mathcal{U}(P_{X_1}, P_{X_2})$).*

Definition 3 (Wasserstein Barycenter)**.** *Given a set of input distributions $(P_{X_1}, \cdots, P_{X_M})$ defined on some space $\mathcal{X}$, weights $\boldsymbol{w}$ such that $\sum_m w_m = 1$, the barycenter is defined as:* $\text{bary}(P_{X_1}, P_{X_2}, \cdots, P_{X_M}; \boldsymbol{w}) \triangleq \arg\min_{P_{X_{\text{bary}}}} \sum_{m=1}^M w_m \mathcal{L}_c(P_{X_{\text{bary}}}, P_{X_m})$, *where $\mathcal{L}_c$ is defined in Def. 2.*

Finally, we review the Wasserstein-2 distance between distributions that will be the basis for the tractable sliced Wasserstein distance described next.

Definition 4 (Wasserstein-2 Distance)**.** *The Wasserstein-2 distance is simply $W_2(P_{X_1}, P_{X_2}) = \mathcal{L}_c(P_{X_1}, P_{X_2})^{\frac{1}{2}}$, where $\mathcal{L}_c$ is defined as above and $c(x, y) = \|x - y\|_2^2$.*

2.2 Maximum K-sliced Wasserstein Distance

While in general the Wasserstein-2 distance requires solving a complex optimization problem, in 1D, the distance can be computed in closed-form because the Monge map is known in closed-form. Thus, several works (e.g., Bonneel et al., 2015; Kolouri et al., 2016; Deshpande et al., 2018) propose to use the sliced Wasserstein distance defined as: $\text{SW}(P_{X_1}, P_{X_2}) \triangleq \mathbb{E}_\theta[W_2(P_{\theta^T X_1}, P_{\theta^T X_2})]$ where θ is distributed as a uniform distribution over all unit norm vectors. A variant called the maximum sliced Wasserstein distance has also been proposed $\text{max-SW}(P_{X_1}, P_{X_2}) \triangleq \max_\theta W_2(P_{\theta^T X_1}, P_{\theta^T X_2})$, which is computed along the direciton with the largest W_2 distance (Kolouri et al., 2019). Recently, Dai and Seljak (2021) proposed the maximum K-sliced Wasserstein distance (which they prove is a true metric between distributions) that finds the K orthogonal directions that maximize the W_2 distance along each projection, i.e., $\text{max-K-SW}(P_{X_1}, P_{X_2}) \triangleq \max_{\theta_1,\ldots,\theta_K} \sum_{k=1}^K W_2(P_{\theta_k^T X_1}, P_{\theta_k^T X_2})$ such that $\theta_k^T \theta_{k'} = 0, \forall k \neq k'$ and $\|\theta_k\|_2 = 1$.

3 MULTIPLE DISTRIBUTION ALIGNMENT

To handle multi-distribution alignment, we first define a symmetric Monge map problem and show that the solution is related to the barycenter problem. This new multi-distribution problem suggests a natural multi-distribution extension to the maximum K-sliced Wasserstein distance, which will be the divergence we seek to minimize in our iterative algorithm.

3.1 Symmetric Monge Map Formulation

The original Monge formulation is asymmetric because the two distributions have distinct roles. While in theory the role of the distributions does not matter because $T^*_{m' \to m} \equiv (T^*_{m \to m'})^{-1}$, in practice the estimated map $\hat{T}$ will vary depending on which distribution is the source distribution. Finally and more importantly, the Monge problem in its original formulation only considers two distributions but we want to consider more than two distributions.

Definition 5 (Symmetric Monge Map (SMM))**.** *Given a set of continuous input distributions $(P_{X_1}, \ldots, P_{X_M})$ defined on some continuous space $\mathcal{X}$, a non-negative weight vector $\boldsymbol{w} \geq 0$ such that $\sum_m w_m = 1$, and cost function $c(\cdot,\cdot)$, the* symmetric Monge map problem *is defined as:*

$$\begin{aligned} \underset{T_1, T_2, \cdots, T_M}{\arg\min} \quad & \sum_{m=1}^M w_m \mathbb{E}_{P_{X_m}} [c(x, T_m(x))] \\ \text{s.t.} \quad & P_{T_m(X_m)} = P_{T_{m'}(X_{m'})} \ \ \forall m \neq m' . \end{aligned} \tag{2}$$

When M=2, the original Monge problem can be recovered if $T_2 = \text{id}$ and $w_2 = 0$. Thus, this problem can be seen as a symmetric relaxation of the Monge problem for two or more distributions. We prove that our symmetric Monge map problem is equivalent to

finding the maps to the barycenter (proof in appendix).

Theorem 1 (SMM Solution is Monge Maps To Barycenter)**.** *For $c(x,y) = \|x-y\|_2^2$ where the distributions have densities, the symmetric Monge map solution (Def. 5) is the Monge maps between the class distributions and the barycenter distribution (Def. 3), i.e., $T_m^* = T_{m\to\text{bary}}^*$ where $P_{X_{\text{bary}}} = \text{bary}(P_{X_1}, P_{X_2}, \ldots, P_{X_M}; \boldsymbol{w})$.*

An important special case where both the barycenter distribution and the Monge maps are known in closed-form is 1D distributions. Thus, in combination with this theorem, we can solve the SMM problem in closed-form for 1D distributions in our iterative algorithm.

3.2 Multiple Distribution Divergences

Just as the Wasserstein distance can be directly derived from the optimum value of the Monge map problem, the optimal value of the SMM problem can provide a natural multi-distribution divergence.

Definition 6. *The multi-distribution Wasserstein divergence is defined as* $\text{Multi-W}(P_{X_1}, \cdots, P_{X_M}) \triangleq \min_{T_1,T_2,\cdots,T_M} \sum_{m=1}^M w_m \mathbb{E}_{P_{X_m}}[c(x, T_m(x))]$ *such that* $P_{T_m(X_m)} = P_{T_{m'}(X_{m'})} \ \forall m \neq m'$.

Similarly, we can define the multi-distribution version of the maximum K-sliced Wasserstein, which we will use to develop our iterative algorithm in the next section.

Definition 7. *The multi-distribution maximum K-sliced Wasserstein divergence is defined as* $\text{Multi-max-K-SW}(P_{X_1}, \cdots, P_{X_M}) \triangleq \max_{\theta_1,\ldots,\theta_K} \sum_{k=1}^K \text{Multi-W}_2(P_{\theta_k^T X_1}, \cdots, P_{\theta_k^T X_M})$.

The proof that these are divergences (i.e., that they are non-negative and have a value of 0 if and only if the distributions are equal) follows easily from the solutions to the SMM problem (see appendix for details).

4 ITERATIVE DISTRIBUTION ALIGNMENT

As a reminder, our ultimate alignment goal is to solve the following problem:

$$\min_{T_1,\cdots,T_M} \phi(P_{T_1(X_1)}, P_{T_2(X_2)}, \cdots, P_{T_M(X_M)}), \quad (3)$$

where ϕ is a multi-distribution divergence. In general a multi-distribution divergence is challenging to even approximate, thus we turn to the sliced Wasserstein versions which are tractable to estimate even for empirical distributions. In particular, the multi-distribution max-K-SW can be written as a maximization problem over a variational approximation of the divergence denoted by $\tilde{\phi}$ and parameterized by $\theta = (\theta_1, \cdots, \theta_K)$, i.e.,

$$\phi(P_{X_1}, \cdots, P_{X_M}) = \max_\theta \tilde{\phi}(\theta, P_{X_1}, \cdots, P_{X_M}) \quad (4)$$

where

$$\tilde{\phi}(\theta, P_{X_1}, \cdots) \triangleq \sum_{k=1}^K \text{Multi-W}_2(P_{\theta_k^T X_1}, \cdots, P_{\theta_k^T X_M}).$$

Importantly, note that $\text{Multi-W}_2(P_{\theta_k^T X_1}, \cdots, P_{\theta_k^T X_M})$ is tractable to compute in closed-form by sorting the data projected onto each direction. Combining Eqn. 3 and Eqn. 4, we arrive at the following min-max optimization for alignment:

$$\min_{T_1,\cdots,T_M} \max_\theta \tilde{\phi}(\theta, P_{T_1(X_1)}, P_{T_2(X_2)}, \cdots, P_{T_M(X_M)}). \quad (5)$$

While this is an adversarial problem, we will not use explicit simultaneous adversarial optimization, which can be challenging as discussed in the introduction. Rather, we derive a simple alternating iterative approach to this problem which is made possible by the tractable structure of our divergence. At a high level, we alternate between solving the inner maximization and the outer minimization. The maximization step forms a variational approximation of the divergence given the current transport maps. The minimization step adds an invertible layer that *globally minimizes* this variational divergence (i.e., where $\tilde{\phi} = 0$). More detailed discussion of the optimization of our algorithms can be found in Appendix C.2. The INB algorithm can be seen in Alg. 2 where for simplicity of exposition, we assume $K = d$ and the more general case is discussed in Appendix G.2.

For the maximization step, we perform gradient descent on the empirical versions of the multi-distribution maximum K-sliced Wasserstein divergence. This objective can be written in closed-form as the following problem:

$$\arg\max_{\boldsymbol{\theta}:\boldsymbol{\theta}^T\boldsymbol{\theta}=I_K} \sum_{m=1}^M \frac{w_m}{K} \sum_{k=1}^K \frac{1}{n_m} \sum_{i=1}^{n_m} |(\boldsymbol{\theta}_k^T \boldsymbol{x}_m)_{[i]} - \boldsymbol{y}_{[i],k}|^2, \quad (6)$$

where $\boldsymbol{x}_m \in \mathbb{R}^{d\times n_m}$ is the sample data matrix for the m-th class, $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K]$, $(\boldsymbol{\theta}_k^T \boldsymbol{x}_m)_{[i]}$ signify the samples from the m-th class distribution projected along the direction $\boldsymbol{\theta}_k$ sorted in ascending order, $\boldsymbol{y}_{[i],k} \triangleq \sum_{m=1}^M w_m (\boldsymbol{\theta}_k^T \boldsymbol{x}_m)_{[i]}$ is the empirical barycenter along direction $\boldsymbol{\theta}_k$, $K \leq d$ is the number of directions, and $I_K \in \mathbb{R}^{K\times K}$ is the identity matrix. Intuitively, this finds the directions that reveal the largest difference between class distributions along each 1D projection. We adopt the optimization approach in Sliced Iterative Normalizing Flows (SINF) that optimizes θ directly

on the manifold of orthonormal matrices (also called a Stiefel manifold) using projected gradient descent with backtracking line search (details in Dai and Seljak (2021)). The algorithm Multi-max-K-SW can be seen in Alg. 3.

The minimization step can be solved exactly via the SMM solutions for 1D distributions (i.e., Monge maps to the barycenter, which is also known in closed-form) by estimating each of the 1D distributions and then solving for the maps. Specifically, the solution would be $T_m^* = F_{\text{bary}}^{-1} \circ F_m$, where F_m is the CDF function of the P_{X_m} distribution and F_{bary}^{-1} is the inverse CDF of the barycenter distribution, which is known to have the following form $F_{\text{bary}}^{-1}(u) = \sum_m w_m F_m^{-1}(u)$. The 1D-Barycenter algorithm can be seen in Alg. 1. These SMM solutions also *locally* minimize the transportation costs to avoid unnecessary distortion from the class distributions. Therefore, the shared latent distributions will be less distorted than if standard generative normalizing flows were used for each distribution independently (see Experiments).

Algorithm 1 1D-Barycenter

Input: Samples from the M class distributions $(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_M)$, weight vector $\boldsymbol{w}$
Output: Estimated invertible alignment maps $(t_1, t_2, \cdots, t_M)$
  for $m = 1, \ldots, M$ **do**
    {Estimate the 1D CDF of Z_m}
    $F_m = \text{HistogramDensityEstimation}(\boldsymbol{z}_m)$
  end for
  {Estimate the inverse CDF of barycenter}
  $F_{\text{bary}}^{-1} = \sum_m w_m F_m^{-1}$
  $\forall m,\ t_m = F_{\text{bary}}^{-1} \circ F_m$
  return $(t_1, t_2, \cdots, t_M)$

Algorithm 2 Iterative Naïve Barycenter Algorithm

Input: Samples from the M class distributions $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M$, weight vector $\boldsymbol{w}$, number of directions K, number of iterations/layers L
Output: Estimated invertible deep alignment maps $(T_1, T_2, \ldots, T_M)$
  $T_m^{(0)} \leftarrow \text{id}, \quad \forall m = \{1, \ldots, M\}$
  for $\ell = \{1, 2, \ldots, L\}$ **do**
    $\forall m, \boldsymbol{z}_m \leftarrow T_m(\boldsymbol{x}_m)$
    {Maximization (see Appendix C for algorithm)}
    $\boldsymbol{\theta} \leftarrow \text{Multi-max-K-SW}((\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M), \boldsymbol{w}, K)$
    {Minimization}
    for $k = \{1, \ldots, K\}$ **do**
      $\forall m,\ \boldsymbol{z}'_m = \boldsymbol{\theta}_k^T \boldsymbol{z}_m$ {1D projection}
      $t_{1,k}, \ldots, t_{M,k} = \text{1D-Barycenter}(\boldsymbol{z}'_1, \ldots, \boldsymbol{z}'_M)$
    end for
    $\forall m, t_m \leftarrow [t_{m,1}, \ldots, t_{m,K}]$
    $\forall m, T_m(\boldsymbol{x}) \leftarrow \boldsymbol{\theta} t_m(\boldsymbol{\theta}^T T_m(\boldsymbol{x}))$
  end for
  return $(T_1, T_2, \cdots, T_M)$

5 RELATED WORK

Iterative Methods Iterative Gaussianization is an iterative density estimation method, that learns invertible flow-based models (Chen and Gopinath, 2000; Lin et al., 2000; Lyu and Simoncelli, 2009; Laparra et al., 2011; Ballé et al., 2016). The key idea is to first learn a rotation matrix via ICA (Hyvarinen, 2013) or similar method to linearly transform the data, and then Gaussianize each marginal independently. Inouye and Ravikumar (2018) extend this by iteratively building normalizing flows from more general "shallow" density estimation approaches. However, these prior iterative approaches are focused on density estimation (i.e., learning a generative model), and in particular, learn a map between a *known* base distribution (e.g. Gaussian) and the unknown data distribution.

Iterative approaches for aligning distributions include Projection Pursuit Monge Map (Meng et al., 2019) that iteratively finds interesting directions to project the data onto, and estimates Monge maps for the 1D projected data. The caveat, however, is that it uses fixed interestingness functions such as variance to find the projection directions. Kuang and Tabak (2019) propose an alternative iterative method for learning optimal maps and the shared representation where each iteration requires the solution of a simpler but *joint* optimal transport problem—rather than solving 1D OT problems as in Meng et al. (2019). In practice, Kuang and Tabak (2019) use a set of fixed interestingness functions to find the needed structure. Essid et al. (2019) extend this iterative approach by using an adversarial objective to automatically learn these interestingness functions. Dai and Seljak (2021) propose SINF as a generative model. In theory, the approach could be used to align two distributions but all the experiments in SINF focus on generative models in which one of the distributions is a Gaussian distribution. They directly solve the optimal transport problem between the source and target distributions. In contrast, our approach constructs the map through a shared distribution which preserve the shared structure; thus distorting the original distributions less. Moreover, thanks to the formulation of barycenter problems, we can naturally deal with multiple distributions, which cannot be done in SINF. For $M > 2$, SINF would need to learn $\binom{M}{2}$ translation maps separately while our model would jointly learn M maps. As we show in the Section 6, our model achieves better alignment performance even for $M = 2$ experiments.

Adversarial Methods CycleGAN (Zhu et al., 2017) minimizes the objective function:

$$\arg\min_{G,F} d_{\text{adv}}(P_{G(X_1)}, P_{X_2}) + d_{\text{adv}}(P_{F(X_2)}, P_{X_1}) + \lambda\Big(\mathbb{E}_{P_{X_1}}[\|F(G(x))-x\|_1]+\mathbb{E}_{P_{X_2}}[\|G(F(x))-x\|_1]\Big),$$

where the distance d_{adv} approximates the distance between distributions via adversarial learning (i.e., minimax learning) and the cycle consistency terms (after λ) can be seen as a relaxation of an invertibility constraint. StarGAN (Choi et al., 2018) generalize CycleGAN to more than two domains. However, these approaches cannot guarantee invertibility and require expensive and challenging adversarial learning (Lucic et al., 2018; Kurach et al., 2019).

Flow Methods AlignFlow (Grover et al., 2020) extends CycleGAN by using invertible models so that the cycle consistency constraint is satisfied by construction:

$$\arg\min_{T_1,T_2} d_{\text{adv}}(P_{T_2^{-1}\circ T_1(X_1)}, P_{X_2})+d_{\text{adv}}(P_{T_1^{-1}\circ T_2(X_2)}, P_{X_1}) + \lambda\Big(\text{KL}(P_{X_1}, P_{T_1^{-1}(\alpha)} + \text{KL}(P_{X_2}, P_{T_2^{-1}(\alpha)}\Big), \tag{7}$$

where the first two distance terms (equivalent to CycleGAN) are implemented using adversarial learning, α is a Gaussian prior distribution, and the KL terms are implemented via maximum likelihood. Unlike our formulation, AlignFlow ignores transportation costs entirely and pushes the shared latent representation towards the assumed prior distribution α rather than the more natural shared latent distribution. Also, for $M > 2$, AlignFlow would require $\binom{M}{2}$ adversarial terms where each term adds significant complexity to training the model.

Wasserstein Barycenter Methods Most existing methods compute Waaserstein Barycenter of discrete distributions. For example, Cuturi and Doucet (2014) propose a fast algorithm with entropic regularization. Those methods typically scale poorly with the number of dimensions and are not suitable for many modern machine learning problems. More recently, several efficient and scalable methods have been proposed to estimate Wasserstein Barycenter over continuous spaces (Li et al., 2020; Chen et al., 2021; Korotin et al., 2021). Chen et al. (2021) utilize input convex neural networks (ICNN) (Amos et al., 2017) to estimate both a generator for barycenter distribution and the transportation map between mariginals and barycenter. A typical difference between our method and those models is that even though we use the solution to 1D barycenter, our main goal is to efficiently learn *invertible flow models* between marginal distributions rather than estimate the global barycenter.

Domain Adaptation Methods Domain Adaptation has become more and more popular recently and there have been several works that leverage the tools from OT for it. Courty et al. (2017b) propose to find the discrete optimal transportation map between source and target domains with class regularization. JDOT (Courty et al., 2017a) improves it by directly aligning the joint distribution of the marginals and the class conditional distributions. DeepJDOT (Damodaran et al., 2018) futher improves JDOT by learning a shared space in a Convolutional Neural Network for classification. All these methods are based on solving discrete OT problems and JDOT and DeepJDOT focus more on finding a classifier for domain adaptation instead of finding invertible alignment maps directly.

6 EXPERIMENTS

We explore our iterative alignment method both qualitatively and quantitatively using both 2D simulated data and "permuted" MNIST (LeCun and Cortes, 2010)—permuted means that our methods do not leverage the image structure of MNIST but merely treat each image as 784-dimensional vector.[1] Additional experiments, implementation details and results can be found in the appendix, including experiments on FashionMNIST (Xiao et al., 2017).

Metrics We use standard distribution distances to compare the alignment performance across methods. We first note that the alignment condition can equivalently be written as $P_{X_m} = P_{T_m^{-1}\circ T_{m'}(X_{m'})}, \quad \forall m \neq m'$. Thus, for every class distribution P_{X_m}, we can sample $M-1$ "fake" distributions using our invertible transformations $\hat{P}_{X_{m'\to m}} = P_{\hat{T}_m^{-1}\circ \hat{T}_{m'}(X_{m'})}$. We merely average the empirical Wasserstein distance between all pairs of real samples and "fake" samples, i.e., $\text{WD} = \frac{1}{M^2-M}\sum_{m\neq m'} \hat{W}(P_{X_m}, \hat{P}_{X_{m'\to m}})$, where $\hat{W}$ is the Wasserstein distance estimated using samples via the Sinkhorn algorithm (Cuturi, 2013) with $\epsilon = 10^{-4}$ and maximum iterations set to 100. For higher dimensional data (e.g., MNIST), the Wasserstein distance between samples could be a poor estimator of the true Wasserstein distance (Genevay et al., 2019). Thus, we also compute the Frechet Inception Distance score (FID) (Heusel et al., 2017) for a more fair evaluation, and we similarly compute the average between every pair of real and fake samples, i.e., $\text{FID} = \frac{1}{M^2-M}\sum_{m\neq m'} \hat{\text{FID}}(P_{X_m}, \hat{P}_{X_{m'\to m}})$. We also compute transportation cost to highlight that our algorithm distorts the distributions less and finds a shared latent distribution that is closer to the original dis-

[1] AlignFlow and the FID we use for evaluation do use the image structure but our iterative methods do not.

tributions because we use the SMM solution for the minimization subproblem, which can be seen to locally minimize the transportation cost. We estimate the transportation cost by an average over the test set, i.e., $\text{TC} = \sum_{m=1}^{M} \frac{w_m}{n_m} \sum_{x \in X_m} \|x - \hat{T}_m(x)\|^2$, where X_m is the test dataset for the m-th class and lower is better. We compute the mean and standard deviation over 5 runs of each method. We also track approximate wall-clock training time for MNIST (all models are trained on a CPU except AlignFlow which is trained on a single GTX 1080 Ti and SINF-Align which is trained on Tesla P100). More details are in the appendix.

Baseline Methods Because prior iterative method focuses on generative models rather than distribution alignment, we adapt prior generative methods to produce alignment approaches. First, we adapt the iterative density destructors method (DD) (Inouye and Ravikumar, 2018) by learning *independent* normalizing flows from each class distribution to a fixed uniform distribution, which is the same for all class distributions and thus serves as a fixed shared latent space. We also adapt SINF (Dai and Seljak, 2021) to the alignment task (SINF-Align) where we directly find the map between two distributions without any shared representation. While the SINF paper mentioned that SINF could be used to align any two distributions, the experiments in the paper assumed that one of the distributions was a standard normal distribution—i.e., only generative experiments were performed. Given that SINF is not symmetric (a point emphasized in the SINF paper), we train two SINFs: one from distribution X_0 to X_1 and the other in the reverse direction. We notice a significant difference of the performance of the forward and inverse of SINF maps. Specifically, the forward map performs well but the inverse map performs poorly (detailed results given in the appendix). These results suggest that the direction of learning is critical and that a symmetric formulation is more stable. For MNIST, as a non-iterative baseline, we compare to the invertible AlignFlow (Grover et al., 2020), which explicitly maps both distributions to an assumed prior distribution.

Our Methods For our methods, except for INB, as a comparison, we also report the results with the single-layer independent (naïve) barycenter (NB) (assume all features are independent of each other and learn alignment maps directly without any projection) and multi-layer random rotation followed by NB (Rand-INB). Number of layers and other parameters are in the appendix.

Computational complexity of INB. The complexity of the maximization is $\mathcal{O}(J(nMK(d+\log n)+K^2d+K^3))$, where J, M, n, d, K are the number of iterations ($J_{\max}$ in Alg. 3), classes, samples per class, dimensions, and latent dimensions, respectively. The terms come from projecting down to K dimensions, computing SWD via sorting, and updating the projection matrix. The complexity of the inner minimization is $\mathcal{O}(nMK)$ since each latent dimension can be computed independently and primarily estimates histograms, which have piecewise linear CDFs and inverse CDFs.

Table 2: Transportation cost (TC), sample-based Wasserstein distance (WD, lower is better) for 2D data. More 2D datasets in appendix.

Model	WD	TC
NB	0.0788 ± 0.0000	**0.4013 ± 0.000**
Rand-NB	0.0047 ± 0.0011	0.4903 ± 0.0205
INB	**0.0025 ± 0.0005**	0.4832 ± 0.0282
DD	0.0085 ± 0.0000	1.2564 ± 0.0000
SINF-Align(0⇒1)	**0.0024 ± 0.0002**	—
SINF-Align(1⇒0)	**0.0026 ± 0.0003**	—

2D Experiments The qualitative results in Fig. 1 illustrate that our method (INB) finds shared latent space where the transportation cost is low (i.e., where the map distorts the original distributions less), whereas density destructors (DD) ignores transportation costs and projects both distributions to the uniform distribution. The results for the 2D datasets with $M = 2$ in Table 2 demonstrate that our iterative flows perform comparable or better than the baseline methods (DD, SINF-Align) in terms of alignment, which is measured by the empirical Wasserstein-2 distance on test data (WD), while having significantly lower transportation cost (TC) on test data. Fig. 1e shows that INB converges much faster than Rand-NB. Additional experiments and results for $M > 2$ are in appendix.

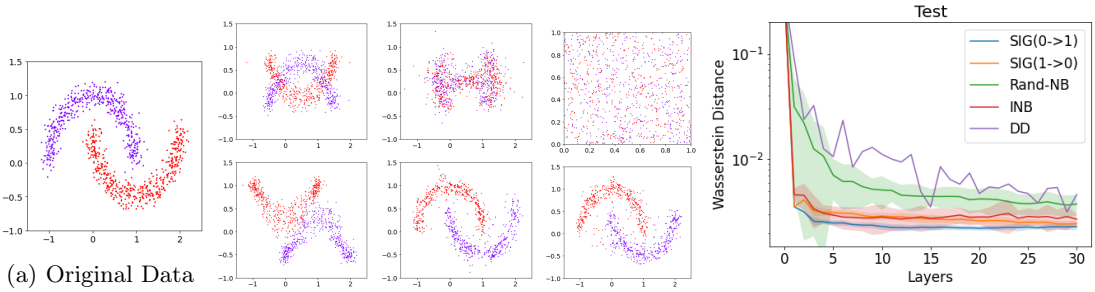
"Permuted" MNIST Qualitative samples from the latent space and after flipping between the two digits (Fig. 2) highlight that our methods retain shared latent structure such as the black pixels, whereas the generative baselines (DD, AlignFlow) move the shared latent distribution to the assumed prior (uniform or Gaussian, respectively) so that shared structure is also removed. Quantitative results in Table 3 demonstrate that INB has superior performance in terms of both WD and FID. Regarding SINF, because the original paper does not test their model on alignment task, we attempt to use their best model to be fair. We report the result of the best SINF-Align models where the number of layers is chosen based on the best test WD. Note that SINF-Align usually achieves the best WD after a few layers and that is why the time we report is quite short. The results demonstrate that our methods perform well in terms of the alignment condition (measured by WD and FID where lower is better) than the

Table 3: Transportation cost (TC), sample-based Wasserstein distance (WD, lower is better), FID (lower is better) and time for MNIST($M = 2$). For a fair comparison, the K used for INB ($L = 20$) is adjusted to be the same as SINF which is 56.

Model	WD	FID	TC	Time(s)
NB	60.010 ± 0.000	229.551 ± 0.000	**28.115 ± 0.000**	**25**
INB ($L = 20$)	23.481 ± 0.161	43.196 ± 0.588	31.671 ± 0.056	430
INB ($L = 250$)	**23.183 ± 0.095**	**37.480 ± 0.008**	32.841 ± 0.097	2200
DD	39.079 ± 0.000	166.320 ± 0.000	235.164 ± 0.000	360
SINF-Align($0 \Rightarrow 1$)	50.151 ± 0.950	247.142 ± 0.972	—	50
SINF-Align($1 \Rightarrow 0$)	42.658 ± 1.253	202.058± 1.716	—	50
AlignFlow(λ =1e-4)	56.386	158.654	392.578	220000
AlignFlow(λ =1e-5)	60.452	191.983	412.531	220000

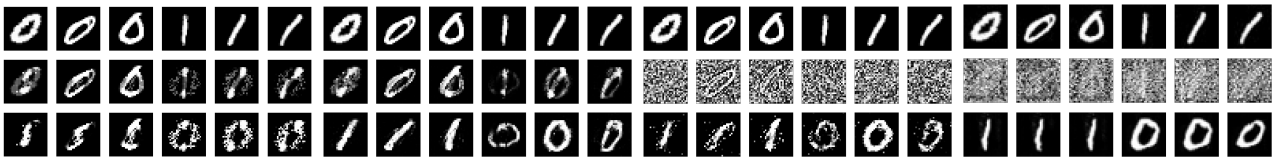
Table 4: Transportation cost (TC), sample-based Wasserstein distance (WD, lower is better), FID (lower is better) and time for MNIST($M = 10$).

Model	WD	FID	TC	Time(s)
NB	65.674 ± 0.000	190.920 ± 0.000	**25.907 ± 0.000**	**90**
INB	**41.044 ± 0.076**	**86.264 ± 0.550**	28.934 ± 0.140	5000
DD	53.587 ± 0.000	187.475 ± 0.000	227.171 ± 0.000	1700



(a) Original Data (b) NB (c) INB (d) DD (e) WD over 30 Layers

Figure 1: The purple and red moons in Fig. 1a represent distributions P_{X_1} and P_{X_2}. The goal is to flip them (i.e. find $P_{X_1'} = P_{T^*_{2\to1}(X_2)}$ and $P_{X_2'} = P_{T^*_{1\to2}(X_1)}$). The shared representations (top row) for each method show that our iterative methods (INB) find low transportation cost shared latent spaces whereas DD ignores transportation cost and merely projects both distributions to the uniform distribution. The bottom row shows test samples that were flipped to the other class distribution (ideally these "fake" samples would look like the original data). Fig. 1e shows that INB converge faster than Rand-NB because we optimize for the directions. SIG represents SINF-Align with SIG setup (details in the appendix).



(a) NB (b) INB (c) DD (d) AlignFlow

Figure 2: These examples demonstrate that our methods find a more natural shared latent representation that preserves structural similarities (e.g., black pixels) between the two digits while DD and AlignFlow do not. The rows from top to bottom are the original MNIST digits, their shared latent representation, and their projection to the space of the other digit (i.e., flipped).

Table 5: Multi-distribution ($M = 10$) results for MNIST with INB. The labels of the rows represent the class of real samples and the labels of the columns represent the class of flipped samples e.g. the number in the row "2" and column "4" represents the WD between the real "4" samples and the fake "4" flipped from "2" samples.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0.10	13.37	48.57	42.77	35.85	41.80	36.13	30.86	45.42	32.10
1	39.86	0.49	47.85	41.98	34.91	41.60	34.78	29.59	44.08	31.01
2	41.02	13.86	0.05	43.88	36.75	43.93	37.43	31.24	46.24	33.47
3	40.80	13.55	48.66	0.04	36.94	43.38	36.44	31.32	45.81	33.22
4	40.63	14.12	48.89	43.24	0.09	43.37	37.22	31.68	46.12	32.89
5	40.61	13.46	49.03	42.95	36.39	0.08	36.20	31.42	45.66	32.88
6	40.31	13.84	48.87	42.56	36.26	41.57	0.11	30.19	44.88	32.18
7	40.08	13.44	48.40	42.42	36.58	42.43	35.57	0.14	45.69	32.91
8	40.59	13.67	49.48	44.32	37.94	43.41	36.47	32.01	0.08	34.12
9	40.16	13.88	48.44	43.36	35.54	42.42	36.26	30.97	45.51	0.14

iterative baseline (DD, SINF-Align) and end-to-end baseline AlignFlow. Also, the computational cost is much lower for the iterative methods ($<$ 1 hour on CPU when $M = 2$), whereas AlignFlow trained for 200 epochs on a GPU took approximately 60 hours (thus, we only estimate one model and cannot compute standard deviations for AlignFlow).

While we use $M = 2$ to fairly compare to prior methods, our method focuses on multi-distribution alignment for $M > 2$. Therefore, we present quantitative results for $M = 10$ in Table 4 and Table 5 (results for $M = 3$ in the appendix). Because no prior methods consider the multi-class case, we only show DD as a baseline method which learns M independent flows to the uniform. This multi-class situation (i.e. $M > 2$) is much more difficult for AlignFlow (which did not implement $M > 2$) and would naïvely require $(M^2 - M)/2$ pairwise adversarial loss terms. SINF does not provide any natural way to handle the $M > 2$ case as well. Qualitative examples of transforming between every digit and every other digit (i.e., $M = 10$) for MNIST are shown in Fig. 3. Notice that even for this multiclass case, almost all transformed digits are recognizable. We observe that the distributions are not fully aligned in the shared space (e.g., some digit structure remains). On one hand, this could explain why there are artifacts in the flipped samples in some cases (e.g., "6"). On the other hand, we hypothesize that this indicates the smoothness of our method and explains why it has better alignment compared to SINF-Align. We leave further investigation to future works.

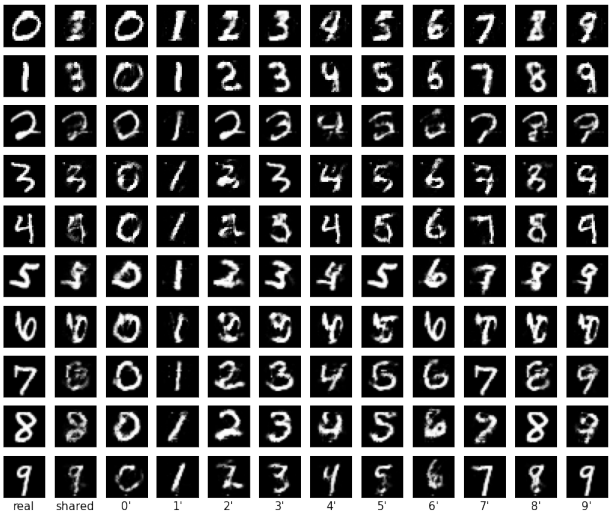


Figure 3: Multi-distribution ($M = 10$) results for MNIST with INB. The first column shows the real samples and the second column shows their shared latent representations. The following columns show the mappings of the real samples to the distribution of the other digits e.g. all flipped samples in the first row are flipped from the real 0 in the first column.

7 DISCUSSION AND CONCLUSION

We seek to iteratively align multiple distributions without adversarial learning. We leverage insights from OT theory to construct an iterative estimation algorithm that alternates between estimation of a tractable divergence via maximization and exact minimization of this variational divergence. Unlike prior approaches, our formulation does not require a fixed latent distribution and can be symmetrically applied to any number of distributions. Unlike prior approaches based on deep normalizing flows, our approach is significantly faster. Despite many advantages of our approach, however, there are also many open challenges. For example, our current algorithm is greedy. Though its greedy nature makes it easy to implement, we cannot guarantee that it finds the globally optimal alignment solution. We leave exploring the non-greedy algorithm to future work. We believe our work is a first step towards non-adversarial distribution alignment that can open up novel perspectives on distribution alignment.

Acknowledgements

D.I., Z.Z., and Z.G. acknowledge support from the Army Research Lab through contract number W911NF-2020-221. P. R. acknowledges the support of NSF via IIS-1909816.

References

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

Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, volume 70, pages 146–155, 2017.

Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. Density modeling of images using a generalized normalization transformation. In *International Conference on Learning Representations, ICLR*, 2016.

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and radon wasserstein barycenters of measures. *J. Math. Imaging Vis.*, pages 22–45, 2015.

Scott Shaobing Chen and Ramesh A. Gopinath. Gaussianization. *NIPS*, pages 423–429, 2000.

Yongxin Chen, Jiaojiao Fan, and Amirhossein Taghvaei. Scalable computations of wasserstein barycenter via input convex neural networks. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, pages 1571–1581, 2021.

Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8789–8797, 2018.

Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 3730–3739, 2017a.

Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1853–1865, 2017b.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, pages 2292–2300, 2013.

Marco Cuturi and Arnaud Doucet. Fast computation of wasserstein barycenters. In *Proceedings of the 31th International Conference on Machine Learning, ICML*, 2014.

Biwei Dai and Uros Seljak. Sliced iterative normalizing flows. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, 2021.

Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *ECCV*, pages 467–483, 2018.

Ishan Deshpande, Ziyu Zhang, and Alexander G. Schwing. Generative modeling using the sliced wasserstein distance. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 3483–3491, 2018.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR*, 2015.

Montacer Essid, Debra F Laefer, and Esteban G Tabak. Adaptive optimal transport. *Information and Inference: A Journal of the IMA*, 8(4):789–816, 2019.

Aude Genevay, Lénaic Chizat, Francis Bach, Marco Cuturi, and Gabriel Peyré. Sample complexity of sinkhorn divergences, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

Aditya Grover, Christopher Chute, Rui Shu, Zhangjie Cao, and Stefano Ermon. Alignflow: Cycle consistent learning from multiple domains via normalizing flows. In *AAAI*, pages 4028–4035, 2020.

Laleh Haghverdi, Aaron TL Lun, Michael D Morgan, and John C Marioni. Batch effects in single-cell rna-sequencing data are corrected by matching mutual nearest neighbors. *Nature biotechnology*, 36(5):421–427, 2018.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

Lanqing Hu, Meina Kan, Shiguang Shan, and Xilin Chen. Duplex generative adversarial network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1498–1507, June 2018.

Aapo Hyvarinen. Independent component analysis: recent advances. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984), 2013.

David I. Inouye and Pradeep Ravikumar. Deep density destructors. In *International Conference on Machine Learning, ICML*, pages 2172–2180, 2018.

Soheil Kolouri, Yang Zou, and Gustavo K. Rohde. Sliced wasserstein kernels for probability distributions. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 5258–5267, 2016.

Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo K. Rohde. Generalized sliced wasserstein distances. In *Advances in Neural Information Processing Systems*, pages 261–272, 2019.

Alexander Korotin, Lingxiao Li, Justin Solomon, and Evgeny Burnaev. Continuous wasserstein-2 barycenter estimation without minimax optimization. In *9th International Conference on Learning Representations, ICLR*, 2021.

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

Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The GAN landscape: Losses, architectures, regularization, and normalization, 2019. URL `https://openreview.net/forum?id=rkGG6s0qKQ`.

Valero Laparra, Gustavo Camps-Valls, and Jesús Malo. Iterative Gaussianization: From ICA to random rotations. *IEEE Transactions on Neural Networks*, 22 (4):537–549, 2011.

Tam Le, Viet Huynh, Nhat Ho, Dinh Phung, and Makoto Yamada. Tree-wasserstein barycenter for large-scale multilevel clustering and scalable bayes. *arXiv preprint arXiv:1910.04483*, 2019a.

Tam Le, Makoto Yamada, Kenji Fukumizu, and Marco Cuturi. Tree-sliced variants of wasserstein distances. In *Advances in Neural Information Processing Systems*, pages 12283–12294, 2019b.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Lingxiao Li, Aude Genevay, Mikhail Yurochkin, and Justin M. Solomon. Continuous regularized wasserstein barycenters. In *Advances in Neural Information Processing Systems 33*, 2020.

J.J. Lin, N. Saito, and R.A. Levine. An iterative nonlinear Gaussianization algorithm for resampling dependent components. *Proc. 2nd International Workshop on Independent Component Analysis and Blind Signal Separation*, pages 245–250, 2000.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. In *Advances in neural information processing systems*, pages 700–709, 2018.

Siwei Lyu and Eero P Simoncelli. Nonlinear extraction of independent components of natural images using radial Gaussianization. *Neural computation*, 21:1485–1519, 2009.

Cheng Meng, Yuan Ke, Jingyi Zhang, Mengrui Zhang, Wenxuan Zhong, and Ping Ma. Large-scale optimal transport map estimation using projection pursuit. In *Advances in Neural Information Processing Systems*, pages 8118–8129. 2019.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends© in Machine Learning*, 11(5-6):355–607, 2019. ISSN 1935-8237. doi: 10.1561/2200000073. URL `http://dx.doi.org/10.1561/2200000073`.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1530–1538, 2015.

Ben Usman, Avneesh Sud, Nick Dufour, and Kate Saenko. Log-likelihood ratio minimizing flows: Towards robust and quantifiable neural distribution alignment. In *Advances in Neural Information Processing Systems*, pages 21118–21129, 2020.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *International Conference on Machine Learning*, pages 325–333, 2013.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision, ICCV*, pages 2242–2251, 2017.

Supplementary Material: Iterative Alignment Flows

A OVERVIEW

We have organized our appendix as follows:

- Appendix B includes the proofs (and key OT results needed for the proofs).
- Appendix C includes the algorithm of multi-distribution max-K-SW and the discussion of the optimization of our algorithm.
- Appendix D describes an alternative to max-K-SW using tree-sliced Wasserstein divergence instead that could be used within our algorithmic general framework.
- Appendix E describes our investigation on directly using SINF for alignment task.
- Appendix F describes additional experiments including additional FashionMNIST experiments and includes quantitative result tables for these experiments (qualitative figures are included in the final appendix section).
- Appendix G provides more details on our experimental setup including dataset preparation, models, and metric details.
- Appendix H provides both expanded figures from the main paper and new result figures for the additional experiments.

B PROOFS

B.1 Symmetric Monge Map Solution Proofs

Proof of Theorem 1. First, let us denote $P_{X_{\text{bary}}} \triangleq P_{T_m^*(X_m)}$ for any m since they are all equal because of the pushforward condition (at this point we do not assume anything about $P_{X_{\text{bary}}}$). We can prove that T_m^* is the optimal Monge map (which is unique for quadratic cost) from P_{X_m} to $P_{X_{\text{bary}}}$ for all m, i.e., $T_m^* = T_{m\to\text{bary}}^*$, via contradiction. Suppose $T^* \neq T_{m\to\text{bary}}^*$, then T^* could be replaced by the optimal Monge map and the minimum value could be reduced—which is a contradiction to the optimality of T^*. Given this fact and Brenier's theorem (Peyré and Cuturi, 2019, Theorem 2.1) on the equivalence between the Kantorovich and the Monge map problems, we can now transform our original objective at the optimum T_m^* to the Kantorovich barycenter objective from Def. 3 at its optimum:

$$\sum_{m=1}^{M} w_m \mathbb{E}_{P_{X_m}} \left[c(x, T_m^*(x))\right] \tag{8}$$

$$= \sum_{m=1}^{M} w_m \mathbb{E}_{Q_m^*}[c(x_m, x_{\text{bary}})] \tag{9}$$

$$= \sum_{m=1}^{M} w_m \min_{Q_m \in \mathcal{U}(P_{X_m}, P_{X_{\text{bary}}})} \mathbb{E}_{Q_m}[c(x_m, x_{\text{bary}})] \tag{10}$$

$$= \sum_{m=1}^{M} w_m \mathcal{L}_c(P_{X_m}, P_{X_{\text{bary}}}), \tag{11}$$

where the first equality is by Brenier's theorem and Q_m^* is the optimal Kantorovich joint distribution, the second equality is by the definition of the Kantorovich problem, and the third equality is by the definition of $\mathcal{L}_c$. Thus, our objective can be equivalently written as optimizing over $P_{X_{\text{bary}}}$ for the objective above, which is exactly the definition of a barycenter in Def. 3. Thus, $P_{X_{\text{bary}}} = \text{bary}(\mu_1, \mu_2, \cdots \mu_M; \boldsymbol{w})$. □

Proposition 2 (Univariate Barycenter (Peyré and Cuturi, 2019, Remark 9.6))**.** *Given a weight vector $\boldsymbol{w}$ with cost $c(x,y) = \|x-y\|^2$, the inverse CDF of the barycenter is the weighted average inverse CDF of the class distributions, i.e.,*

$$\forall u \in [0,1], \quad F^{-1}_{\text{bary}}(u) = \sum_{m=1}^{M} w_m F_m^{-1}(u)\,, \tag{12}$$

where F_m^{-1} is the inverse CDF of the m-th class distribution.

Proposition 3 (Univariate Optimal Transport Map (Peyré and Cuturi, 2019, Remark 2.30))**.** *The optimal map between univariate distributions P_{X_1} and P_{X_2} is the composition of the CDF of P_{X_1} with the inverse CDF of P_{X_2}, i.e.,*

$$T^*_{1\to 2} = F_2^{-1} \circ F_1\,. \tag{13}$$

Theorem 4 (Optimal 1D Symmetric Monge Maps)**.** *The optimal univariate symmetric Monge maps are: $T^*_m = F^{-1}_{\text{bary}} \circ F_m$, where F_m is the CDF function of the P_{X_m} distribution and F^{-1}_{bary} is the inverse CDF of the barycenter distribution, which is known to have the following form $F^{-1}_{\text{bary}}(u) = \sum_m w_m F_m^{-1}(u)$.*

Proof. From Theorem 1, we know that the solution to the symmetric Monge problem is the Monge map between the class distribution and the barycenter distribution. From Proposition 2, we can form the univariate barycenter distribution given the class distributions. We can then combine this result with Proposition 3 to solve for the optimal map between the univariate class distribution and the univariate barycenter distribution. □

B.2 Divergence Proofs

Proposition 5. Multi-W$(P_{X_1}, \cdots, P_{X_M}) \triangleq \min_{T_1,T_2,\cdots,T_M} \sum_{m=1}^{M} w_m \mathbb{E}_{P_{X_m}}\left[c(x, T_m(x))\right]$ *such that $P_{T_m(X_m)} = P_{T_{m'}(X_{m'})}\ \forall m \neq m'$ as defined in Def. 6 is a divergence.*

Proof. We need to prove two properties: 1) Multi-W$(P_{X_1}, \cdots, P_{X_M}) \geq 0$, and 2) Multi-W$(P_{X_1}, \cdots, P_{X_M}) = 0$ if and only if $P_{X_M} = P_{X_{M'}}, \forall m \neq m'$. The first property is obvious by inspection of the objective function which is always non-negative.

If $P_{X_M} = P_{X_{M'}}, \forall m \neq m'$, then we can use the trivial solution of all maps being the identity, i.e., $\forall m, T_m(\boldsymbol{x}) = \boldsymbol{x}$. By construction, the constraint is satisfied and the cost will be 0, which is the global optimum of the minimization.

If Multi-W$(P_{X_1}, \cdots, P_{X_M}) = 0$, then we know that $\forall \boldsymbol{x}$ and $\forall m$, $c(\boldsymbol{x}, T_m(\boldsymbol{x})) = 0$ (by contradiction if one of them was > 0 then it would violate the assumption that the sum was 0). The only function that satisfies this property would be the identity functions for all T_m. By the constraint of the optimization, we know that $P_{T_m(X_m)} = P_{T_{m'}(X_{m'})}\ \forall m \neq m'$ and thus since these must be the identity, then we know that $P_{X_m} = P_{X_{m'}}, \forall m \neq m'$. □

Proposition 6. Multi-max-K-SW$(P_{X_1}, \cdots, P_{X_M}) \triangleq \max_{\theta_1,\ldots,\theta_K} \sum_{k=1}^{K}$ Multi-W$_2(P_{\theta_k^T X_1}, \cdots, P_{\theta_k^T X_M})$ *as defined in Def. 7 is a divergence.*

Proof. The non-negativity property follows directly from the fact that Multi-W$_2$ is a divergence which is non-negative. We now prove that Multi-max-K-SW$(P_{X_1}, \cdots, P_{X_M}) = 0$ if and only if $P_{X_1} = P_{X_2} = \cdots = P_{X_M}$.

If Multi-max-K-SW$(P_{X_1}, \cdots, P_{X_M}) = 0$, then we can prove that $\forall \theta \in \{\theta \in \mathbb{R}^d : \|\theta\|_2 = 1\}$, Multi-W$_2(P_{\theta^T X_1}, \cdots, P_{\theta^T X_M}) = 0$. (The proof for this statement is by contradiction. Suppose $\exists \theta$ such that Multi-W$_2(P_{\theta^T X_1}, \cdots, P_{\theta^T X_M}) > 0$. Then, we could set $\theta_1 = \theta$ in the maximization problem and Multi-max-K-SW$(P_{X_1}, \cdots, P_{X_M}) > 0$. Yet this is a contradiction to our assumption that Multi-max-K-SW$(P_{X_1}, \cdots, P_{X_M}) = 0$.) Thus, by Proposition 5, we know that $\forall \theta \in \{\theta \in \mathbb{R}^d : \|\theta\|_2 = 1\}, \forall m \neq m', P_{\theta^T X_m} = P_{\theta^T X_{m'}}$. From this we can conclude that $\forall m \neq m', P_{X_m} = P_{X_{m'}}$ because two joint distributions are equal if and only if the marginals along every direction are equal.

If $P_{X_1} = P_{X_2} = \cdots = P_{X_M}$, then we know that the marginals along all directions must be equal, i.e., $\forall \theta \in \{\theta \in \mathbb{R}^d : \|\theta\|_2 = 1\}, \forall m \neq m', P_{\theta^T X_m} = P_{\theta^T X_{m'}}$. Thus, $\forall \theta$, Multi-W$_2(P_{\theta^T X_1}, \cdots, P_{\theta^T X_M}) = 0$ and the maximal value of $\max_{\theta_1,\cdots,\theta_K} \sum_{k=1}^{K}$ Multi-W$_2(P_{\theta_k^T X_1}, \cdots, P_{\theta_k^T X_M})$ must also be 0 for any $\theta_1, \cdots, \theta_K$. Thus, Multi-max-K-SW$(P_{X_1}, \cdots, P_{X_M}) = 0$. □

C ALGORITHMS

C.1 Multi-distribution Maximum K-Sliced Wasserstein Distance

Algorithm 3 Multi-max-K-SW

Input: Samples from the M class distributions $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M)$, weight vector $\boldsymbol{w}$, number of directions K, max number of iterations $J_{\max}$
Output: Estimated projection matrix $\boldsymbol{\theta}$
Randomly initialize $\boldsymbol{\theta} \in \mathbb{R}^{d\times K}$ satisfying $\boldsymbol{\theta}^T\boldsymbol{\theta} = I_K$, $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K]$
for $j = \{1, 2, \ldots, J_{\max}\}$ **do**
  $\boldsymbol{d} = \sum_{m=1}^{M} \frac{w_m}{K} \sum_{k=1}^{K} \frac{1}{n_m} \sum_{i=1}^{n_m} |(\boldsymbol{\theta}_k^T \boldsymbol{x}_m)_{[i]} - \boldsymbol{y}_{[i],k}|^2$
  $\boldsymbol{g} = [-\frac{\partial \boldsymbol{d}}{\partial \boldsymbol{\theta}_{i,j}}]$, $\boldsymbol{u} = [\boldsymbol{g}, \boldsymbol{\theta}]$, $\boldsymbol{v} = [\boldsymbol{\theta}, -\boldsymbol{g}]$
  $\boldsymbol{\theta} = \boldsymbol{\theta} - \tau\boldsymbol{u}(I_{2K} + \frac{\tau}{2}\boldsymbol{v}^T\boldsymbol{u})^{-1}\boldsymbol{v}^T\boldsymbol{\theta}$, learning rate τ determined by backtracking line search
  if $\boldsymbol{\theta}$ converge **then**
    Stop
  end if
end for
return $\boldsymbol{\theta}$

C.2 Discussion of INB

Expanding Eqn. 5 and simplifying to a single slice, i.e., $\theta \in \mathbb{R}^d$

$$
\begin{aligned}
&\min_{T_1,\cdots,T_M} \max_{\theta} \tilde{\phi}(\theta, P_{T_1(X_1)}, P_{T_2(X_2)}, \cdots, P_{T_M(X_M)}) \\
=&\min_{T_1,\cdots,T_M} \max_{\theta} \text{Multi-W}(P_{\theta^T T_1(X_1)}, P_{\theta^T T_2(X_2)}, \cdots, P_{\theta^T T_M(X_M)}) \\
=&\min_{T_1,\cdots,T_M} \max_{\theta} \left(\begin{array}{l} \min\limits_{f_1,\cdots,f_M} \sum\limits_{m=1}^{M} w_m \mathbb{E}_{z\sim P_{\theta^T T_m(X_m)}}[c(z, f_m(z)] \\ \text{s.t.} \quad P_{f_m(\theta^T T_m(X_m))} = P_{f_{m'}(\theta^T T_{m'}(X_{m'}))} \ \ \forall m \neq m' \end{array} \right)
\end{aligned} \tag{14}
$$

At each iteration, for the maximization problem, we use Alg. 3 to find θ. To solve the outer minimization of T_m, we update our transformation with one layer to achieve the global minimum (given the current θ), i.e., $T'_m = t_m \circ T_m$ will be the optimal solution where we construct t_m based on solutions to the inner optimization. Specifically, we solve the inner 1D problems (given a fixed θ and T_m) denoted by $f_1^*, \cdots, f_M^*$ by estimating the 1D CDF function for each class (k in total) using the whole dataset and finding the *local 1D* barycenter map.

$$
\begin{aligned}
f_1^*, \cdots, f_M^* = \underset{f_1,\cdots,f_M}{\arg\min} \sum_{m=1}^{M} w_m \mathbb{E}_{z\sim P_{\theta^T Z_m}}[c(z, f_m(z)] \\
\text{s.t.} \quad P_{f_m(\theta^T Z_m)} = P_{f_{m'}(\theta^T Z_{m'})} \ \ \forall m \neq m'
\end{aligned} \tag{15}
$$

where $Z_m \triangleq T_m(X_m)$. Then, we can construct $\forall m, t_m^*(\boldsymbol{z}) = \theta f_m^*\left(\theta^T \boldsymbol{z}\right) + \left(\boldsymbol{z} - \theta\theta^T \boldsymbol{z}\right)$ as discussed in Appendix G.2. Given a fixed θ, the updated $T'_m = t_m^* \circ T_m$ is the optimal solution to the outer minimization problem (even when the inner minimization is unconstrained).

Proof of optimality. First, note that the new random variable projected along the slice is equal to the transformed 1D slice distribution, i.e.,

$$
\begin{aligned}
\theta^T T'_m(X_m) &= \theta^T t_m^* \circ T_m(X_m) \\
&= \theta^T t_m^*(Z_m) \\
&= \theta^T \left(\theta f_m^*\left(\theta^T Z_m\right) + \left(Z_m - \theta\theta^T Z_m\right)\right) \\
&= f_m^*\left(\theta^T Z_m\right) + \theta^T Z_m - \theta^T Z_m \\
&= f_m^*\left(\theta^T Z_m\right)
\end{aligned}
$$

Now $P_{f_m^*(\theta^T Z_m)} = P_{f_{m'}^*(\theta^T Z_{m'})}, \forall m \neq m'$ by the alignment constraint on the f_m^*'s, and thus, combining with above, we have that $P_{\theta^T T'_m(X_m)} = P_{\theta^T T'_{m'}(X_{m'})}, \forall m \neq m'$. Thus, the W_2 distance along all slices is 0, and T'_m is the globally optimal solution per the property of W_2.

D TREE-SLICED WASSERSTEIN DIVERGENCE

We note that our general variational algorithm could work for other variational tractable divergences such as the tree-sliced Wasserstein (tree-SW) distance (Le et al., 2019b).Because the tree-SW can be seen as a generalization of the SW distance, we could similarly define a max-tree-SW distance and a multi-distribution max-tree-SW divergence. The maximization would be over the tree split structure rather than orthogonal directions as in the multi-distribution maximum K-sliced Wasserstein divergence. Note that the optimal Monge maps for tree-SW are known in closed form similar to the 1D case (Le et al., 2019b). Additionally, the barycenter is also known in closed-form (Le et al., 2019a).Thus, the inner maximization problem over tree structures could use a decision tree algorithm to approximately solve the inner maximization problem. The outer minimization could be solved by first finding the barycenter in closed-form and then computing the optimal maps to this barycenter in closed-form. For this last step, the tree-SW only provides the amount of mass to move between nodes but does not explicitly define the continuous invertible function to do so. For simplicity, we can assume the distribution has support on the unit hypercube (if it does not, then we can use CDF functions of the appropriate marginal distributions so that it does satisfy this constraint). For each movement, we could merely define a piecewise linear function defined over the unit interval to move mass across the split. This could be defined in a top down fashion where at the root node, we use a piecewise linear function to move mass from the left to the right of the split and then recursively apply this idea the nodes below. This would create a piecewise linear invertible function over continuous space that would match the optimal tree Monge maps.

E FAILURE OF SINF FOR ALIGNMENT TASK

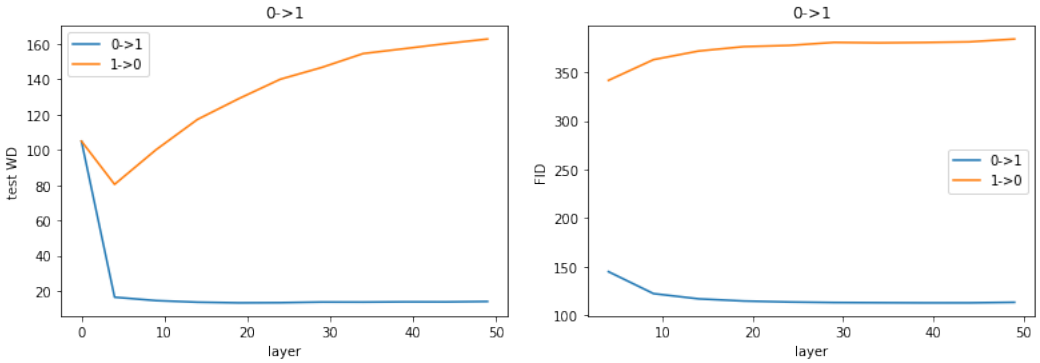
In Dai and Seljak (2021), they propose Sliced Iterative Normalizing Flows (SINF). SINF first projects the data into lower dimensional space using orthogonal projection found by max-K-SWD. Then it aligns the distribution along each direction using know solution to 1D OT problems. Though they state that this could be used to find the transformation between any two distributions, in the paper, they fix one of the distribution to be standard normal distribution. In specific, they propose Sliced Iterative Generator (SIG) and Gaussianizing Iterative Slicing (GIS) and they report that the two models perform better for generative modeling and density estimation separately.

In this paper, we report the results of SINF-Align($0 \Rightarrow 1$) and SINF-Align($1 \Rightarrow 0$). When reporting WD and FID, since SINF is an invertible model, we use the inverse of SINF for inverse transformation and then compute the average. Since they don't provide any result of applying their model for alignment task, we try our best to compare fairly - we use the same $K = 56$ as what they set as default value for MNIST and FashionMNIST and we don't include any hierarchical structure. And we report the test results at the layer where SINF achieves best test WD.

We observe that for both WD and FID, SINF performs well in the test for task in the same direction of training. In most cases, it converges quite fast and is relatively stable. However, when we use it for inverse task, the result is very bad. In most cases, the WD and FID would keep increasing as we add more layers. See Figure 4 and Figure 5 for qualitative results. We want to emphasize the possible failure of directly using SINF for inverse task. In contrast, our model is trained based on a symmetric objective which naturally avoids this problem.

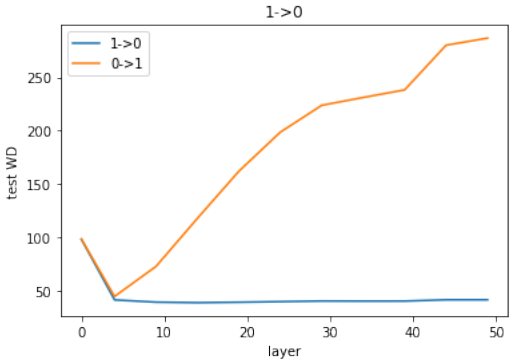
F ADDITIONAL EXPERIMENTS

In this section, we include the quantitative results for all experiments in addition to those presented in the main paper. In the following subsections, brief introductions of each experiment are provided. More experiment details are provided in the next section.

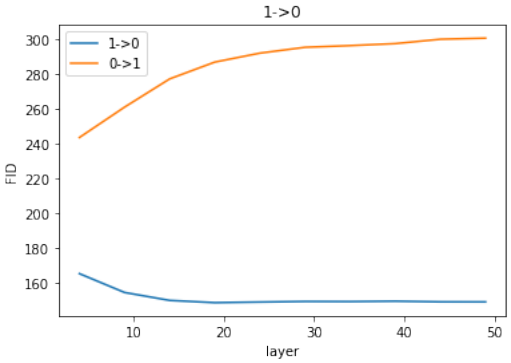


(a) WD: SINF($0 \Rightarrow 1$)

(b) FID: SINF($0 \Rightarrow 1$)



(c) WD: SINF($1 \Rightarrow 0$)



(d) FID: SINF($1 \Rightarrow 0$)

Figure 4: Results of SINF-Align for MNIST($M = 2$). The results are recorded after each 5 layers. The label of the curve represents which task it is used for.

F.1 2D Experiment for All Datasets

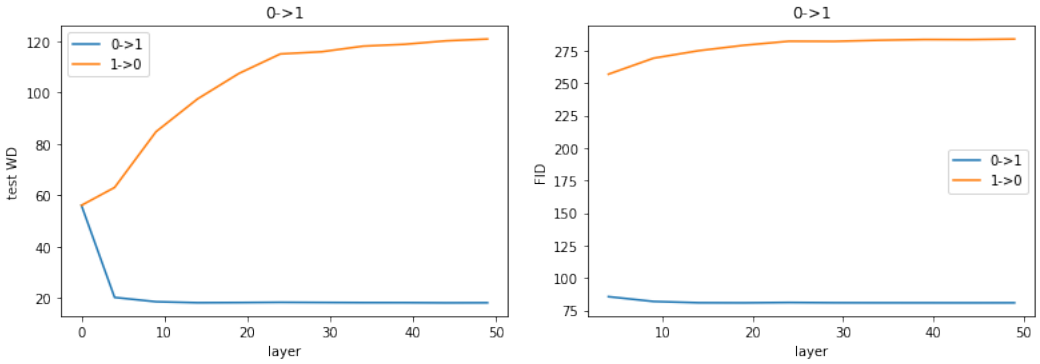
For the 2D datasets with $M = 2$, we investigate the performance of our iterative methods along with the baselines DD and SINF-Align. For $M > 2$, we only compare to DD since SINF does not have a natural extension for multiple distributions. See Table 6 and Table 7 for quantitative results. See Figure 6, Figure 17, Figure 18 and Figure 19 for expanded figures of the latent representation and translations between distributions. In both $M = 2$ and $M > 2$ cases, INB successfully translates the distributions to look similar to the original data (i.e., the fake distributions by translating from one class to another are similar to the original distributions).

F.2 FashionMNIST with $M = 2$ Class Distributions

We redo the experiment for MNIST with $M = 2$ in the main paper for FashionMNIST with $M = 2$. See Table 8 for quantitative result. See Figure 7 and Figure 8 for expanded figures of MNIST and FashionMNIST. For fairness, we simply pick the first three samples in test set here. These examples demonstrate that our methods find a more parsimonious shared latent representation that preserves structural similarities (e.g., black pixels) between the two digits while DD and AlignFlow do not preserve this shared structure.

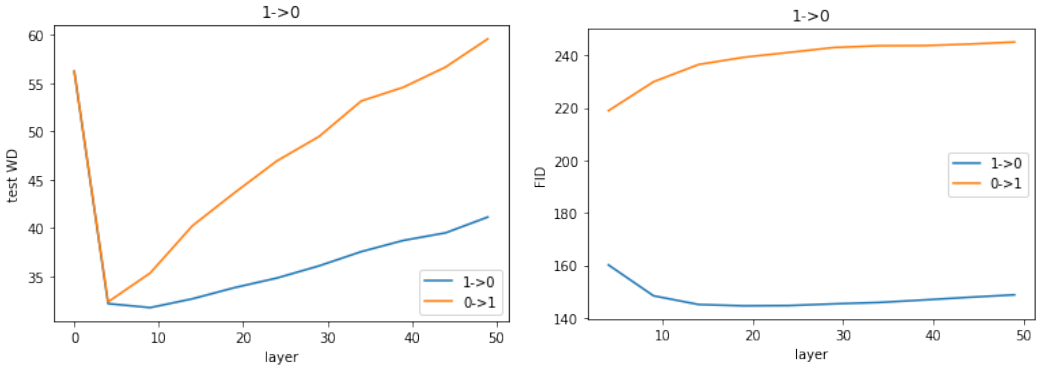
F.3 MNIST and FashionMNIST with $M = 3$ Class Distributions

We investigate the performance of our models together with DD for more than two class distributions. See Table 9 for quantitative result. See Figure 9 and Figure 10 for mapping performance. For fairness, we pick the first sample in test set here. The latent representation of our models keeps more features of original samples while DD just projects to uniform distribution.



(a) WD: SINF-Align($0 \Rightarrow 1$)

(b) FID: SINF-Align($0 \Rightarrow 1$)



(c) WD: SINF-Align($1 \Rightarrow 0$)

(d) FID: SINF-Align($1 \Rightarrow 0$)

Figure 5: Results of SINF-Align for FashionMNIST($M = 2$). The results are recorded after each 5 layers. The label of the curve represents which task it is used for.

F.4 MNIST and FashionMNIST with $M = 10$ Class Distributions

We investigate the performance of our models together with DD for ten class distributions. See Table 10 for quantitative result. See Table 11, Table 12, Table 13, Table 14, Table 15, Table 16 for WD table for each digit with different models. See Figure 11, Figure 12, Figure 13, Figure 14, Figure 15, Figure 16, for expanded figures of mapping performance with different models. For fairness, we pick the first sample in test set here. We can observe that with INB, most mappings seem good though the model struggles to translate in certain cases such as from 6 to 8.

Table 6: Transportation cost (TC), sample-based Wasserstein distance (WD, lower is better) for 2D data. The best methods (within one standard deviation of the top method) are bolded.

	Moon		Random Pattern		Circles	
Model	**WD**	**TC**	**WD**	**TC**	**WD**	**TC**
NB	0.0788 ± 0.0000	**0.4013 ± 0.000**	0.3173 ± 0.0000	**0.9537 ± 0.0000**	0.0042 ± 0.0000	**0.0602 ± 0.0000**
Rand-NB	0.0047 ± 0.0011	0.4903 ± 0.0205	0.0620 ± 0.0188	1.0234 ± 0.0583	0.0043 ± 0.0015	0.0830 ± 0.0065
INB	**0.0025 ± 0.0005**	0.4832 ± 0.0282	0.0458 ± 0.0260	1.0207 ± 0.0270	0.0033 ± 0.0005	0.0834 ± 0.0090
DD	0.0085 ± 0.0000	1.2564 ± 0.0000	0.0469 ± 0.0000	3.7005 ± 0.0000	**0.0029 ± 0.0000**	1.2580 ± 0.0000
SINF-Align($0 \Rightarrow 1$)	**0.0024 ± 0.0002**	—	**0.0340 ± 0.0083**	—	**0.0028 ± 0.0002**	—
SINF-Align($1 \Rightarrow 0$)	**0.0026 ± 0.0003**	—	0.0637 ± 0.0105	—	**0.0029 ± 0.0002**	—

Table 7: The results for the 2D random pattern dataset with $M = 4$ and 2D Gaussian with $M = 3$ demonstrate that our methods still perform well for $M > 2$ in terms of the pushforward constraint, which is measured by the empirical Wasserstein-2 distance on test data (WD). The best methods (within one standard deviation of the top method) are bolded.

	Random Pattern ($M = 4$)		**Gaussian** ($M = 3$)	
Model	**WD**	**TC**	**WD**	**TC**
NB	0.488 ± 0.000	**9.084 ± 0.000**	0.692 ± 0.000	**7.027 ± 0.000**
Rand-NB	**0.155 ± 0.023**	9.652 ± 0.094	**0.067 ± 0.001**	7.469 ± 0.018
INB	**0.153 ± 0.023**	9.532 ± 0.062	**0.065 ± 0.002**	7.461 ± 0.006
DD	**0.154 ± 0.000**	9.434 ± 0.000	0.096 ± 0.000	7.851 ± 0.000

Table 8: Results for FashionMNIST with $M = 2$. The best methods (within one standard deviation of the top method) are bolded.

Model	**WD**	**FID**	**TC**	**Time(s)**
NB	44.038 ± 0.000	118.285 ± 0.000	**20.522 ± 0.000**	**40**
INB ($L = 20$)	24.976 ± 0.092	84.802 ± 0.744	25.964 ± 0.122	430
INB ($L = 250$)	**24.553 ± 0.129**	**79.829 ± 0.928**	26.989 ± 0.060	2800
DD	27.913 ± 0.000	90.546 ± 0.000	181.401 ± 0.000	300
SINF-Align($0 \Rightarrow 1$)	41.111 ± 0.800	169.722 ± 1.452	—	50
SINF-Align($1 \Rightarrow 0$)	31.897 ± 0.184	187.153 ± 0.670	—	50

Table 9: Results for MNIST and FashionMNIST with $M = 3$. It shows that our method enables a natural extension beyond the two class case without requiring a significant increase in computational complexity. The best methods (within one standard deviation of the top method) are bolded. INB used for FashionMNIST is set to be $L = 100$ and $K = 10$.

Dataset	**MNIST**($M = 3$)				**FashionMNIST**($M = 3$)			
Model	**WD**	**FID**	**TC**	**Time(s)**	**WD**	**FID**	**TC**	**Time(s)**
NB	84.408 ± 0.000	229.778 ± 0.000	**28.958 ± 0.000**	**25**	71.341 ± 0.000	166.114 ± 0.000	**28.233 ± 0.000**	**25**
INB	**40.116 ± 0.115**	**158.940 ± 0.695**	34.062 ± 0.090	3700	**41.820 ± 0.142**	**116.871 ± 1.615**	34.374 ± 0.077	1400
DD	60.226 ± 0.000	220.308 ± 0.000	233.354 ± 0.000	320	44.975 ± 0.000	131.043 ± 0.000	171.150 ± 0.000	470

Table 10: Transportation cost (TC), sample-based Wasserstein distance (WD, lower is better), FID (lower is better) and time for MNIST and FashionMNIST($M = 10$).

Dataset	**MNIST**($M = 10$)				**FashionMNIST**($M = 10$)			
Model	**WD**	**FID**	**TC**	**Time(s)**	**WD**	**FID**	**TC**	**Time(s)**
NB	65.674 ± 0.000	190.920 ± 0.000	**25.907 ± 0.000**	**90**	60.288 ± 0.000	172.690 ± 0.000	**47.272 ± 0.000**	**90**
INB	**41.044 ± 0.076**	**86.264 ± 0.550**	28.934 ± 0.140	5000	**36.439 ± 0.042**	**122.619 ± 0.714**	55.128 ± 0.043	5000
DD	53.587 ± 0.000	187.475 ± 0.000	227.171 ± 0.000	1700	40.788 ± 0.000	126.625 ± 0.000	127.099 ± 0.000	1560

Table 11: Multi-distribution ($M = 10$) results for MNIST with INB. The labels of the rows represent the class of real samples and the labels of the columns represent the class of flipped samples e.g. the number in the row "2" and column "4" represents the WD between the real "4" samples and the fake "4" samples flipped from "2" samples.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0.10	13.37	48.57	42.77	35.85	41.80	36.13	30.86	45.42	32.10
1	39.86	0.49	47.85	41.98	34.91	41.60	34.78	29.59	44.08	31.01
2	41.02	13.86	0.05	43.88	36.75	43.93	37.43	31.24	46.24	33.47
3	40.80	13.55	48.66	0.04	36.94	43.38	36.44	31.32	45.81	33.22
4	40.63	14.12	48.89	43.24	0.09	43.37	37.22	31.68	46.12	32.89
5	40.61	13.46	49.03	42.95	36.39	0.08	36.20	31.42	45.66	32.88
6	40.31	13.84	48.87	42.56	36.26	41.57	0.11	30.19	44.88	32.18
7	40.08	13.44	48.40	42.42	36.58	42.43	35.57	0.14	45.69	32.91
8	40.59	13.67	49.48	44.32	37.94	43.41	36.47	32.01	0.08	34.12
9	40.16	13.88	48.44	43.36	35.54	42.42	36.26	30.97	45.51	0.14

Table 12: Multi-distribution ($M = 10$) results for MNIST with NB.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0.05	36.62	68.93	57.06	58.73	57.39	55.07	57.05	60.75	53.41
1	84.16	0.19	79.58	69.92	61.35	76.99	69.73	58.62	71.06	62.15
2	69.26	34.74	0.03	57.43	55.72	69.63	64.54	50.98	61.80	54.20
3	66.71	34.93	66.25	0.02	60.91	56.30	62.10	56.63	60.02	56.26
4	72.76	30.54	71.53	66.01	0.03	68.37	61.77	45.46	62.11	39.27
5	62.54	35.27	73.39	51.94	58.95	0.01	59.55	55.85	57.19	52.40
6	63.69	33.64	72.76	66.62	55.87	65.76	0.06	59.08	64.91	55.18
7	78.07	32.02	72.60	68.39	52.64	71.43	69.12	0.05	66.52	45.02
8	67.04	35.10	68.20	57.73	55.24	56.15	61.91	53.14	0.03	50.75
9	71.66	33.81	71.99	65.97	42.25	65.93	65.09	41.49	60.93	0.04

Table 13: Multi-distribution ($M = 10$) results for MNIST with DD.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0.03	23.55	60.27	50.89	47.45	52.91	48.72	42.76	55.22	44.07
1	54.44	0.01	62.29	52.96	48.32	56.41	49.56	42.67	57.63	44.82
2	53.24	23.62	0.02	51.56	47.54	56.63	49.92	42.09	56.10	44.33
3	53.00	24.22	59.51	0.02	48.86	52.72	49.47	43.34	56.60	45.37
4	52.82	23.33	60.51	53.61	0.01	56.03	48.97	40.75	55.78	39.24
5	51.55	23.49	60.56	50.33	48.01	0.01	48.29	43.18	53.75	44.84
6	52.91	24.27	60.67	53.00	47.38	55.15	0.02	42.67	55.76	44.55
7	54.21	24.35	60.74	53.18	45.79	55.52	49.59	0.02	55.39	41.46
8	52.09	24.09	59.91	52.25	47.30	51.70	49.06	42.69	0.02	44.11
9	53.38	24.41	60.24	53.74	41.86	54.78	49.67	39.50	55.12	0.02

Table 14: Multi-distribution ($M = 10$) results for FashionMNIST with INB.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0.21	18.78	34.80	26.11	31.48	37.13	34.44	21.70	47.89	31.94
1	33.09	0.94	34.88	28.01	31.20	38.49	35.36	21.96	50.44	32.31
2	32.20	18.42	0.28	26.77	28.10	37.65	33.55	21.74	47.43	32.03
3	35.70	19.97	36.51	0.29	33.18	38.86	37.59	22.15	51.41	33.23
4	32.12	18.20	33.05	26.37	0.28	37.31	34.38	21.25	46.77	31.26
5	37.01	21.38	39.18	31.12	36.19	0.13	41.03	26.57	53.54	38.28
6	31.79	18.76	34.56	26.29	30.19	38.17	0.15	21.89	50.18	32.18
7	34.11	19.72	35.82	29.23	32.75	39.21	37.38	0.24	50.34	33.66
8	34.69	20.41	36.50	28.81	32.87	37.06	36.92	21.34	0.04	33.13
9	33.80	19.93	35.84	28.69	32.11	37.73	36.94	21.83	48.10	0.06

Table 15: Multi-distribution ($M = 10$) results for FashionMNIST with NB.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0.01	31.54	47.74	34.91	43.80	64.02	44.57	37.95	77.73	59.11
1	56.71	0.05	70.61	40.41	62.99	76.58	67.14	45.92	100.12	70.20
2	41.04	38.60	0.00	37.02	29.53	58.41	37.37	38.19	69.86	58.64
3	47.76	28.43	60.22	0.02	54.82	69.33	59.57	40.30	91.44	64.35
4	45.42	36.13	36.38	39.10	0.01	62.69	39.69	39.06	73.08	60.68
5	64.39	50.18	73.13	55.80	68.76	0.02	71.95	31.25	85.69	46.48
6	35.17	31.87	37.16	34.27	33.03	58.02	0.00	37.19	71.92	56.84
7	67.13	51.83	76.76	57.98	70.58	52.46	74.84	0.08	95.52	63.89
8	50.87	43.78	53.48	46.20	46.52	51.57	53.05	29.71	0.00	47.45
9	61.18	47.73	67.24	53.22	62.20	50.29	67.60	31.60	76.36	0.01

Table 16: Multi-distribution ($M = 10$) results for FashionMNIST with DD.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	2.27	22.32	34.02	31.18	34.12	46.94	38.32	25.59	58.34	39.85
1	33.26	0.55	35.24	31.82	36.16	47.57	39.30	25.58	61.59	41.87
2	32.05	21.74	2.32	30.75	32.19	45.94	37.05	24.53	56.22	40.31
3	33.77	24.07	33.27	1.06	34.72	47.59	40.46	26.30	60.18	39.87
4	33.57	22.20	31.69	31.13	1.45	45.99	35.26	24.35	55.70	39.95
5	34.94	23.12	36.14	34.61	35.80	0.01	40.96	25.04	61.08	38.95
6	32.41	22.60	32.93	31.12	32.43	46.53	1.80	24.55	57.87	39.91
7	34.12	24.21	35.72	34.49	36.91	45.84	40.66	0.15	61.25	40.71
8	33.00	22.01	33.58	31.80	34.06	46.79	37.78	24.64	0.97	40.03
9	33.95	23.10	35.69	33.70	36.26	45.88	39.43	24.41	58.87	0.29

G EXPERIMENT DETAILS

G.1 Histogram-based 1D Density Estimators for NB Method

For high flexibility yet low computational cost, we choose to use a histogram-based density estimator for our independent component (naiv̈e) layers (NB) in our experiments. While histograms are generally efficient and reasonable non-parametric estimators, one key drawback is that you must choose the interval for the histogram (e.g., using the minimum and maximum of the data). This can yield odd edge conditions if the interval is not chosen properly. Thus, to avoid this challenge, we first estimate a preprocessing transformation to squeeze the data to the interval $[0, 1]$ and then estimate a histogram on this fixed interval. In particular, we merely use a Gaussian CDF (where the mean and covariance are estimated from the data) to preprocess the data. We then estimate a histogram on the transformed data. This can be seen as an almost trivial 1D normalizing flow where the histogram is a learned base prior distribution and the Gaussian CDF is the flow. We use the code from deep density destructors (Inouye and Ravikumar, 2018) to implement this estimation procedure. Note that this estimation procedure only requires estimating a 1D Gaussian and a 1D histogram—both of which have minimal computational cost.

G.2 Details when the number of target directions is less than the dimensionality ($K < d$)

For the INB layer, if the number of target directions K is less than the dimensionality d, we can define a partial independent components layer that only acts on K directions. From a theoretical viewpoint, we could adjust our estimators as follows:

1. For estimating $\boldsymbol{\theta}$, the other $d - K$ directions of $\boldsymbol{\theta}$ can be filled in with an arbitrary orthonormal subspace.
2. When estimating the independent class distributions, we could assume that the $d - K$ directions have the same distribution for *all* classes.

The first assumption allows us to preserve the full dimensionality of the data when projecting into the latent space. The second assumption implies that the transform along the $d - K$ directions is the identity because all the class distributions are the same, which implies that their barycenter is equal to the class distributions, which implies that the symmetric Monge map is merely the identity function (see Proposition 3). Thus, it can be seen that these assumptions roughly just ignore the $d - K$ directions.

In practice, we do not have to actually create a full orthogonal matrix $\boldsymbol{\theta}$ or estimate the class distributions along the other $d - K$ directions. We can instead use truncated orthogonal matrices (i.e., where the columns are orthogonal but it is not square) and truncated joint transformations. More formally, we can create the following invertible but "truncated" transform to avoid unnecessary computation as is done in (Dai and Seljak, 2021):

$$T_m^*(\boldsymbol{x}) = \boldsymbol{\theta} t_m^*(\boldsymbol{\theta}^T \boldsymbol{x}) + \boldsymbol{x}^\perp = \boldsymbol{\theta} t_m^*(\boldsymbol{\theta}^T \boldsymbol{x}) + (\boldsymbol{x} - \boldsymbol{\theta}\boldsymbol{\theta}^T \boldsymbol{x}), \tag{16}$$

where $t_m^* = [t_{m,1}^*, \ldots, t_{m,K}^*]$ and $\boldsymbol{x}^\perp \triangleq \boldsymbol{x} - \boldsymbol{\theta}\boldsymbol{\theta}^T \boldsymbol{x}$ contains the components that are perpendicular to $\boldsymbol{\theta}$. Note that this transformation is invertible and equivalent to the non-truncated "theoretical" version described above but requires significantly less computation.

G.3 Datasets

In each run of our experiments, we use the same data even for our simulated data (i.e., we use the same random seed for generating the data for each run).

2D distributions For 2D data, we use the fixed samples for each repetition of experiment (i.e., we produce simulated data for all runs rather than producing new simulated data for each run).

- $M = 2$: Datasets of Moon, Random Pattern, Circles are generated by `make_moons`, `make_classification` and `make_circles` in `sklearn.datasets` respectively. The original number of training samples is 2000 and the original number of test samples is 1000.
- $M > 2$: Dataset of Random Pattern ($M = 4$) is generated by `make_classification` in `sklearn.datasets`. The original number of training samples is 2666 and the original number of test samples is 1334. Dataset of Gaussian ($M = 3$) is generated by `MultivariateNormal` in `torch.distributions.multivariate_normal` with different means and covariances. The original number of training samples is 4000 and the original number of test samples is 2000.

MNIST and FashionMNIST We first take the full MNIST dataset (70k samples) and split into training and testing split. The dimensionality of MNIST and FashionMNIST datasets is 784. To ensure all classes have the same number of samples in the training and test split, we take the minimum number of samples over all classes and truncate the samples of all digits to that number. The numbers vary slightly depending on the number of class distributions M and datasets but are approximately 4500 samples per digit for training and 2300 samples per digit for testing (Experiment for MNIST and FashionMNIST with $M = 10$ has approximately 1800 samples per digit for testing).

For our models including DD, we preprocess the data by dequantizing the original data with uniform distribution and dividing by 256 to create a continuous distribution over the unit hypercube. For AlignFlow, the data is further normalized to the range $[-1, 1]$ to serve as the input to the Real-NVP and the GAN discriminator as in the original AlignFlow paper.

See below for the exact classes we use for our experiments.

- MNIST with $M = 2$: We use digit 0 and 1.
- FashionMNIST with $M = 2$: We use T-shirt and trouser.
- MNIST with $M = 3$: We use digit 0, 1 and 9.
- FashionMNIST with $M = 3$: We use T-shirt, trouser and pullover.
- MNIST with $M = 10$: We use digit 0-9.

G.4 Models for 2D Experiments

Two class distributions ($M = 2$)

- Number of layers: All iterative models (including INB, NB-INB, Rand-NB, NB-Rand-NB, DD, SINF-Align) use 15 layers.
- Number of dimensions for orthogonal transformation: We apply orthogonal transformation in the full space with dimension 2, i.e., $K = d = 2$.
- INB: We iteratively fit NB after orthogonal transformation based on max sliced Wasserstein distance. The maximum number of iterations for max-K-SW J_{max} is set to be 200 for all experiments.
- Rand-NB: We iteratively fit NB after random orthogonal transformation found by QR decomposition of a matrix generated by `torch.randn`.
- NB-INB and NB-Rand-NB: We first perform a full-dimensional NB layer and then follow this by 14 iterations of INB/Rand-NB.
- DD: For the univariate histogram density estimator, we use 40 bins and set $\alpha = 1$, which corresponds to the pseudo-counts added to each bin.
- SINF-Align: We use the SIG code from original github repo for the SINF paper (Dai and Seljak, 2021).

More than two class distributions ($M > 2$)

- Basically the setup is very similar to the $M = 2$ case. The differences are listed as below.

- Number of layers: All iterative models (including INB, NB-INB, Rand-NB, NB-Rand-NB, DD) use 30 layers.
- Number of dimensions for orthogonal transformation: We apply orthogonal transformation in the space with dimension 2.
- DD: It is basically the same as the $M = 2$ case but with a different initial destructor. Additionally, we add a normal distribution CDF and inverse CDF as pre and post processing transformations.

G.5 Models for MNIST and FashionMNIST

Two class distributions ($M = 2$)

- Number of dimensions for INB: We use orthogonal transformation with $K = 30$ directions which is much smaller than the ambient dimensions of $d = 784$ similar to the the SINF paper (Dai and Seljak, 2021).
- Number of layers: We use 250 layers for INB and 10 layers for DD. In this way, the product of the number of layers and the number of dimensions while fitting the NB/DD are approximately the same i.e. $250 \times 30 \approx 10 \times 784$.
- INB: We add a normal distribution inverse CDF and CDF at the start and the end of the entire INB model as pre and post processing transformations to project the unit data into the real space for transformation.
- DD: The setup of DD is basically the same as what we use for 2D experiments with $M > 2$ except that we remove the pre and post processing transformations with the normal distribution CDF and inverse CDF since the data is already on the unit hypercube.
- Alignflow: The AlignFlow implementation is done through the direct clone from the Github repository with some modifications on the code and parameters setup. We first follow the AlignFlow paper to have these general parameters getting set up: the batch size is 16, the learning rate is set to a fixed `2e-04`, maximum gradient norm is 10. We further set the `data_constraint` value inside `RealNVP` model to be 0.999998.
  We train 200 epochs for choices of the lambda value `1e-05` and `1e-04`. For the Real-NVP model, the model is a four scale setup. The first three scales contain three checkerboard coupling layers followed by three channelwise coupling layers. Then the data is squeezed and split so that half the data goes to the next scale. For the final scale, we only perform the checkerboard coupling layer four times. The squeeze operation is simply by turning each subvolume $4 \times 4 \times 1$ into the subvolume $1 \times 1 \times 4$. And the splitting operation tries to split the last dimension into two parts. Also within each coupling layer, we parameterize the scale and translate factors by using the ResNet structure with number of blocks equals 4. And the number of channels for the ResNet is set to 32 and gets doubled every time when we switch the coupling layer from checkerboard layer to channelwise layer. For the GAN setup, the discriminator is set to have 5 convolutional layers with kernel size 4 and stride 1. The number of channels is doubled each time when passing to the next layer with the initial value 32 for the generator and 64 for the discriminator.

More than two class distributions ($M = 3$)

- INB: The INB used for FashionMNSIT is set to be $L = 100$ and $K = 10$.
- The setup of other models is exactly the same as the $M = 2$ case.

More than two class distributions ($M = 10$)

- Number of dimensions for INB: We use orthogonal transformation with $K = 10$ to transform the original distribution with dimension $d = 784$.
- Number of layers: We use 100 layers for INB since the working dimension is only $K = 10$ for each layer while for DD we only use 10 layers because the working dimension is $d = 784$. Thus, if we compare the total number of dimension-wise transformations INB has $100 \times 10 = 1000$ transformations while DD can have $784 \times 10 = 7840$ transformations. Nevertheless, INB still performs better in general based on our quantitative and qualitative results in other sections.

G.6 Metrics for 2D Experiments

- *Transportation cost* - We find the averaged squared distance for each class separately and use uniform weight to take the average over all class distributions, i.e.,

$$\frac{1}{M} \sum_{m} \frac{1}{|X_m|} \sum_{x \in X_m} \|x - \hat{T}_m(x)\|_2^2 \,, \tag{17}$$

where $|X_m|$ is the number of samples in the test set for the m-th class distribution and $\hat{T}_m$ are the estimated maps.

- *Wasserstein distance* - For the test samples, we form "fake" samples for each class distribution by using the estimated maps, i.e.,

$$\widetilde{X}_{m' \to m} = \hat{T}_m^{-1}(\hat{T}_{m'}(X_{m'})), \quad \forall m' \neq m, \tag{18}$$

where $\hat{T}_m$ are the estimated maps. We then use the Sinkhorn algorithm (with $\epsilon = 10^{-4}$ and maximum iterations set to 100) to estimate the WD between the real and fake samples over all possible real-fake pairs, i.e.,

$$\frac{1}{M^2 - M} \sum_{m=1}^{M} \sum_{m' \neq m} \text{SinkhornWD}(X_m, \widetilde{X}_{m' \to m}) \,. \tag{19}$$

- *Repetitions* - We repeat the entire map estimation process and metric evaluation 5 times to average over random effects and calculate standard deviations for each method (except AlignFlow).

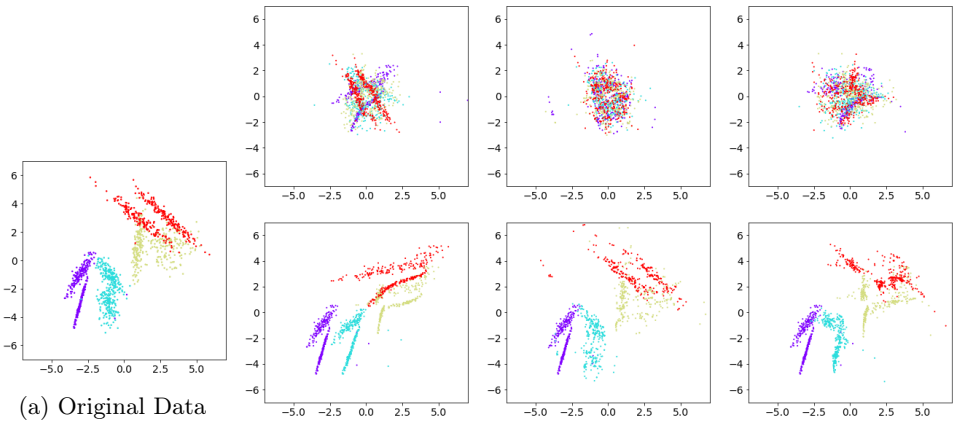
G.7 Metrics for MNIST and FashionMNIST

Transportation Cost The setup for transportation cost is the same as 2D experiments except for the experiment with AlignFlow since the scale of the input and output are different in AlignFlow. Specifically, iterative methods such as NB, INB, and DD, have images and latent spaces to be in the range $[0, 1]$ for each dimension. However, in AlignFlow, images are normalized into $[-1, 1]$, and the latent space is a normal distribution. Therefore, for the purpose of comparison with all the iterative methods, we need some modifications on the transportation cost for the AlignFlow. We can rescale the input domain from $[-1, 1]$ to $[-0.5, 0.5]$ simply by dividing the input by 2, which gives a unit domain as for the iterative methods. We can do the same for the latent space which makes the Gaussian prior to have a standard deviation 0.5 instead of 1. By doing these pre and postprocessing steps, we can get approximately the same scale in both image space and latent space as the unit scale for the iterative methods. The transportation cost is then $c(\frac{1}{2}x, \frac{1}{2}z) = \|\frac{1}{2}(x - z)\|^2 = \frac{1}{4}c(x, z)$. Therefore, we manually divide the transportation cost computed in the unscaled space by a factor of 4 for the AlignFlow paper for the purpose of fair comparison. Note that this added scaling favors the baseline method AlignFlow—without it, the AlignFlow transportation cost would be worse. Additionally, because AlignFlow is so computationally expensive, we do not repeat the estimation process five times and thus cannot compute standard deviation for AlignFlow transportation costs.

Wasserstein Distance The setup of Wasserstein Distance is basically the same as that in 2D experiments except that we partition the data. In all experiments, the partition size is set to be 500. The final WD is computed as the weighted average of that of all partitions.

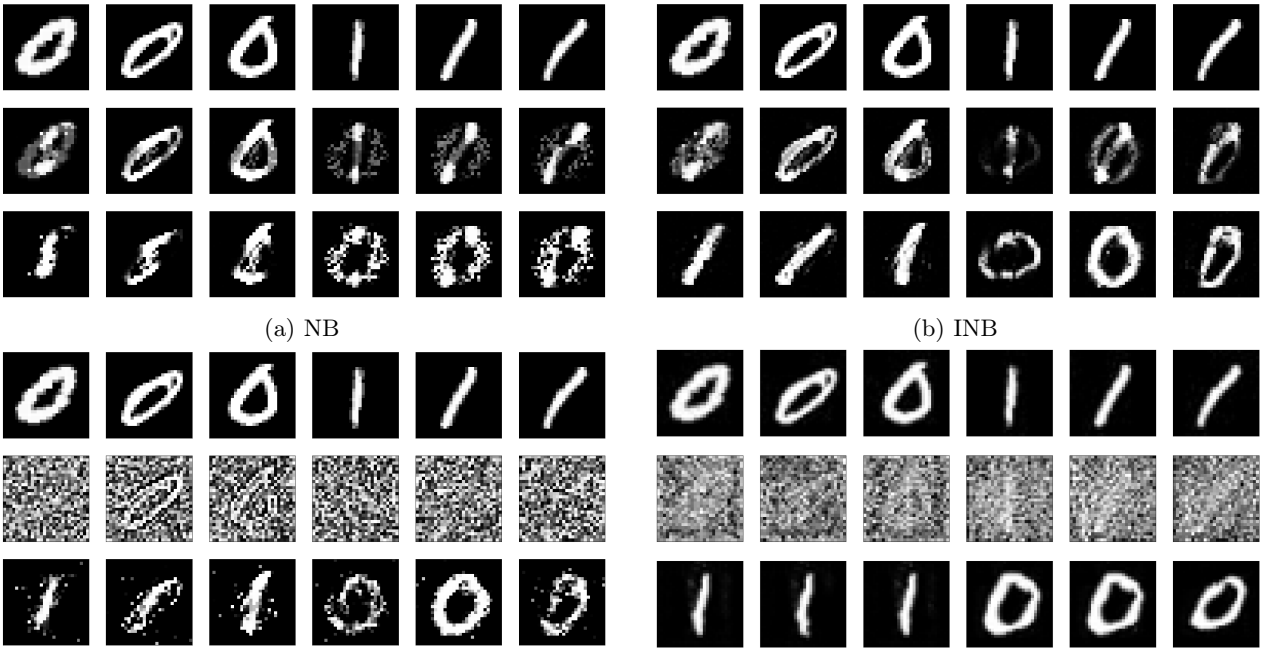
H ADDITIONAL AND EXPANDED FIGURES

This section includes the qualitative results for additional experiments and expanded figures from the main paper.



(a) Original Data (b) NB (c) INB (d) DD

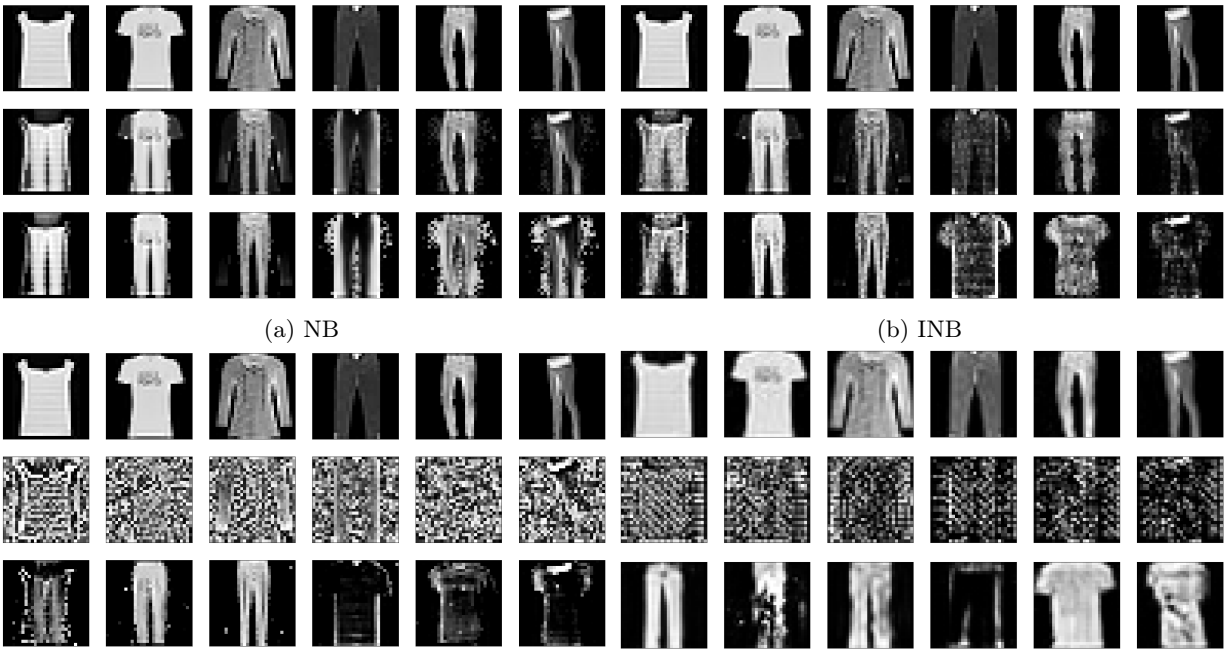
Figure 6: 2D Random Pattern Data ($M = 4$). The top row is the latent distribution found by class 1 data. The bottom row is the corresponding flipped distribution from it.



(a) NB (b) INB

(c) DD (d) AlignFlow

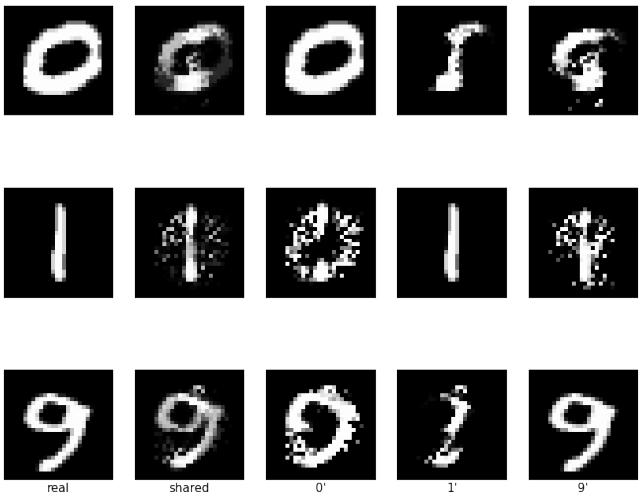
Figure 7: Expanded figure of MNIST ($M = 2$). The first row represents the original samples. The second row represents the latent representation. The third row represents the flipped samples.



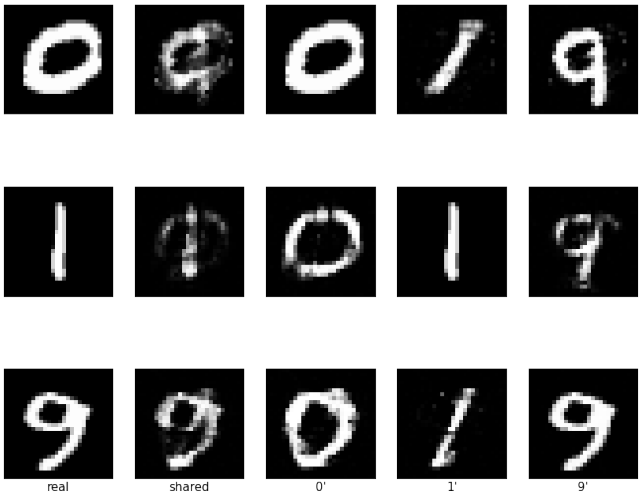
(a) NB (b) INB

(c) DD (d) AlignFlow

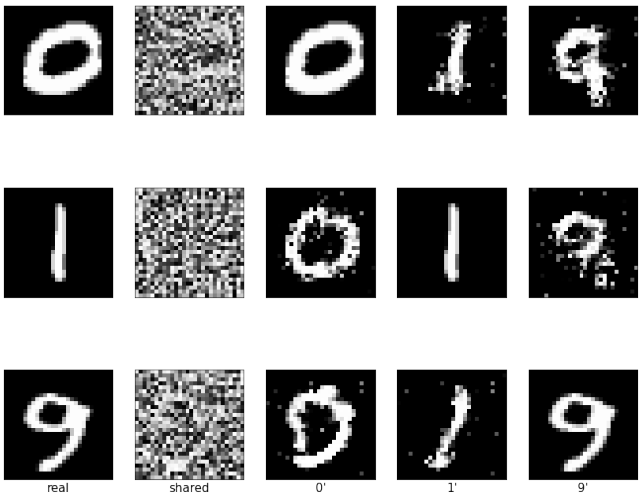
Figure 8: Expanded figure of FashionMNIST ($M = 2$). The first row represents the original samples. The second row represents the latent representation. The third row represents the flipped samples.



(a) NB

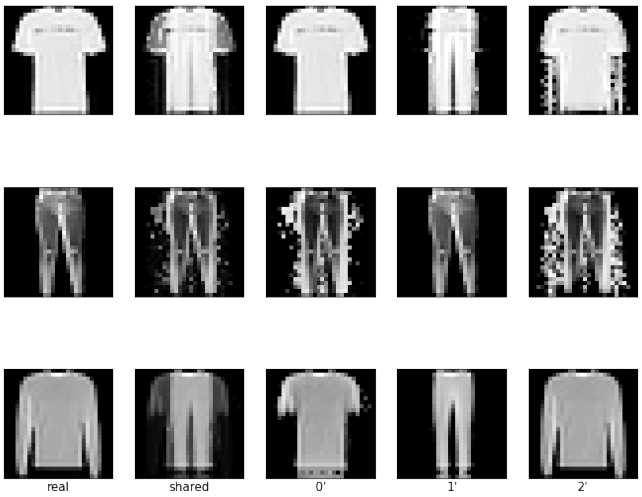


(b) INB

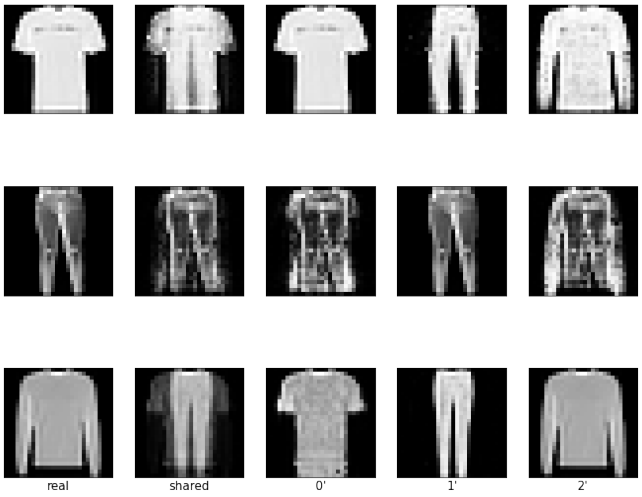


(c) DD

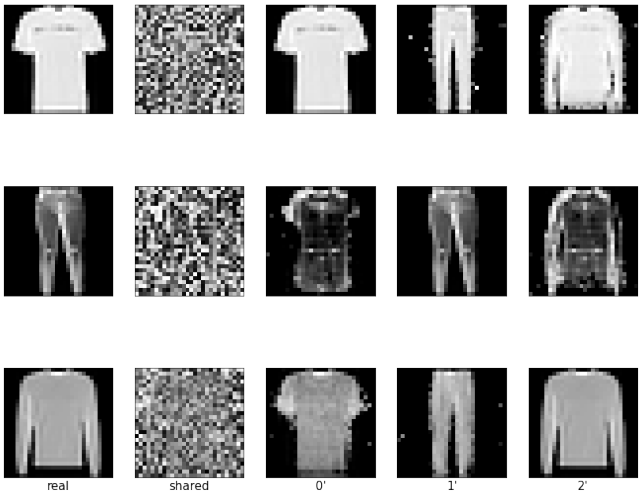
Figure 9: Samples of MNIST ($M = 3$). The first column shows the real samples and the second column shows their shared latent representations. The following columns show the mappings of the real samples to the distribution of the other digits e.g. all flipped samples in the first row are flipped from the real 0 in the first column.



(a) NB



(b) INB



(c) DD

Figure 10: Samples of FashionMNIST ($M = 3$).

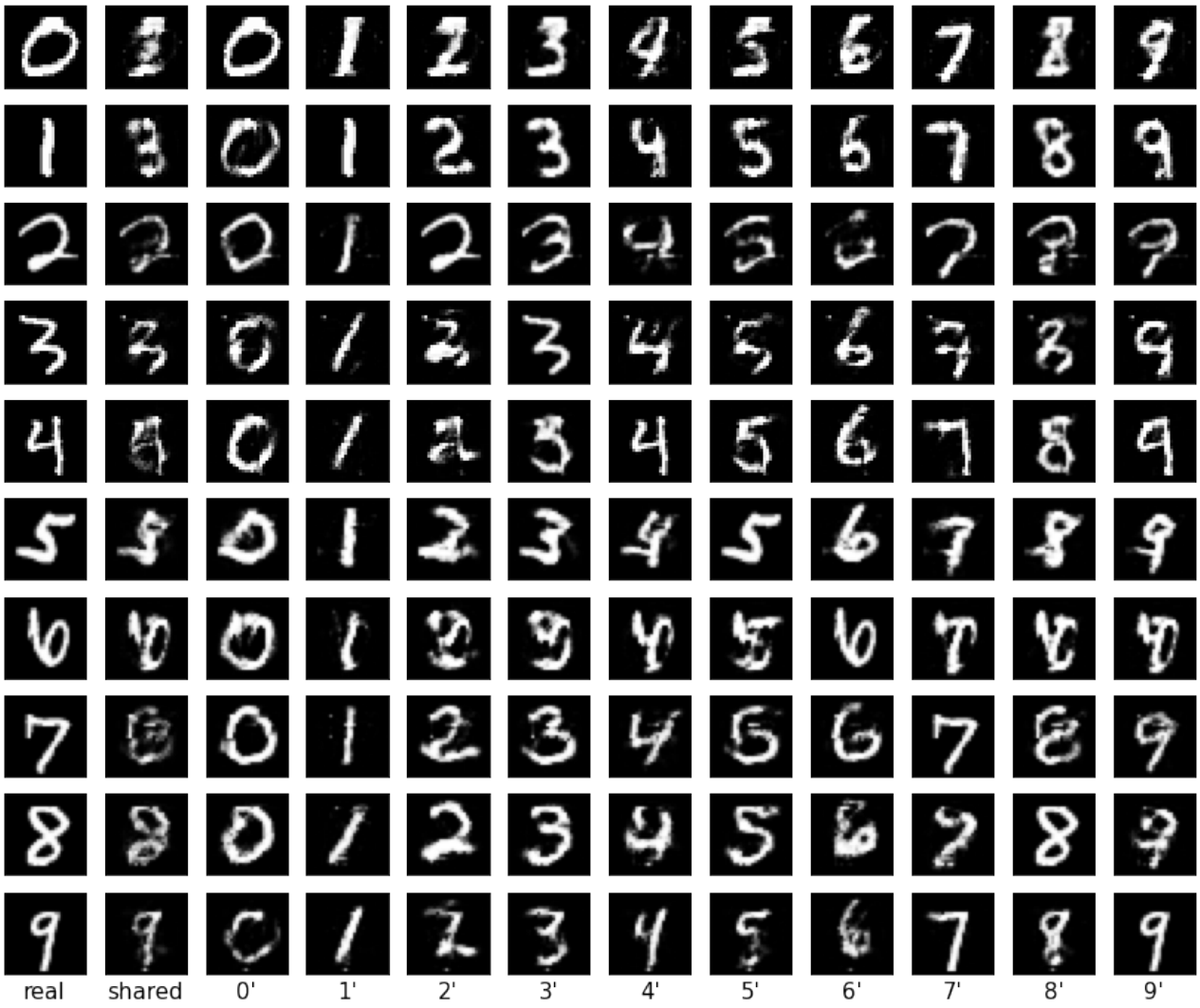


Figure 11: Multi-distribution ($M = 10$) results for MNIST with INB. The first column shows the real samples and the second column shows their shared latent representations. The following columns show the mappings of the real samples to the distribution of the other digits e.g. all flipped samples in the first row are flipped from the real 0 in the first column.

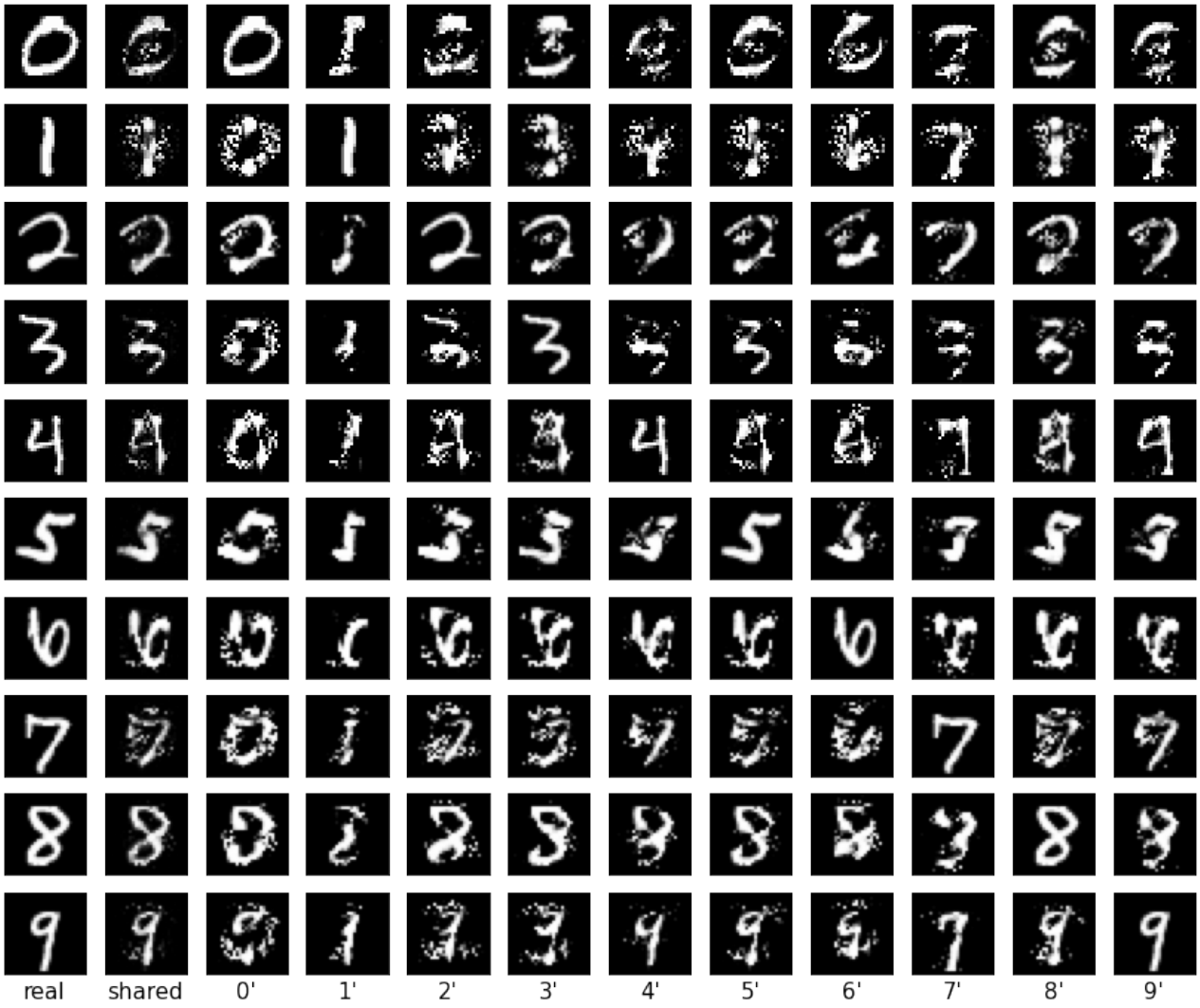


Figure 12: Multi-distribution ($M = 10$) results for MNIST with NB.

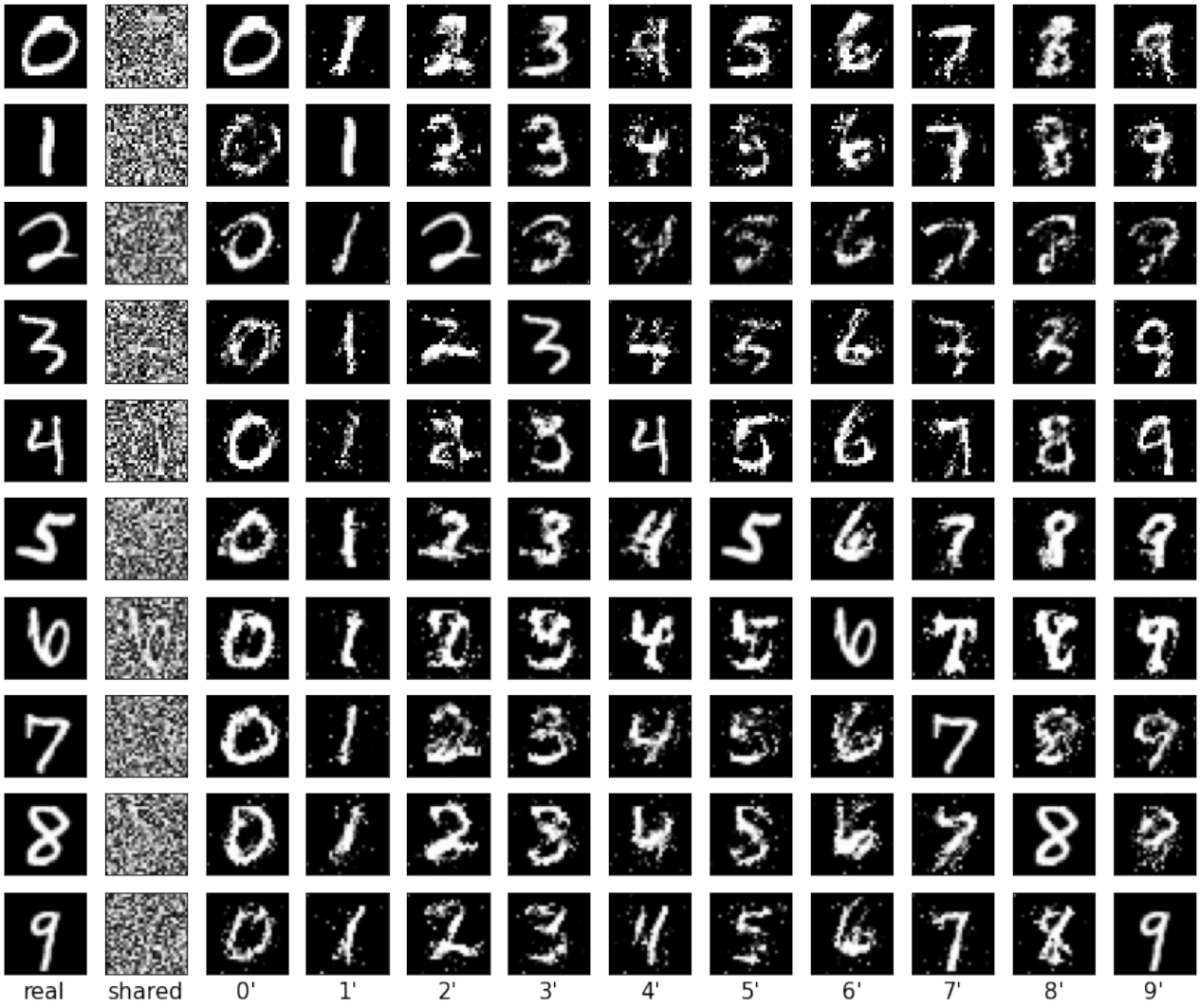


Figure 13: Multi-distribution ($M = 10$) results for MNIST with DD.

Figure 14: Multi-distribution ($M = 10$) results for FashionMNIST with INB.

Figure 15: Multi-distribution ($M = 10$) results for FashionMNIST with NB.

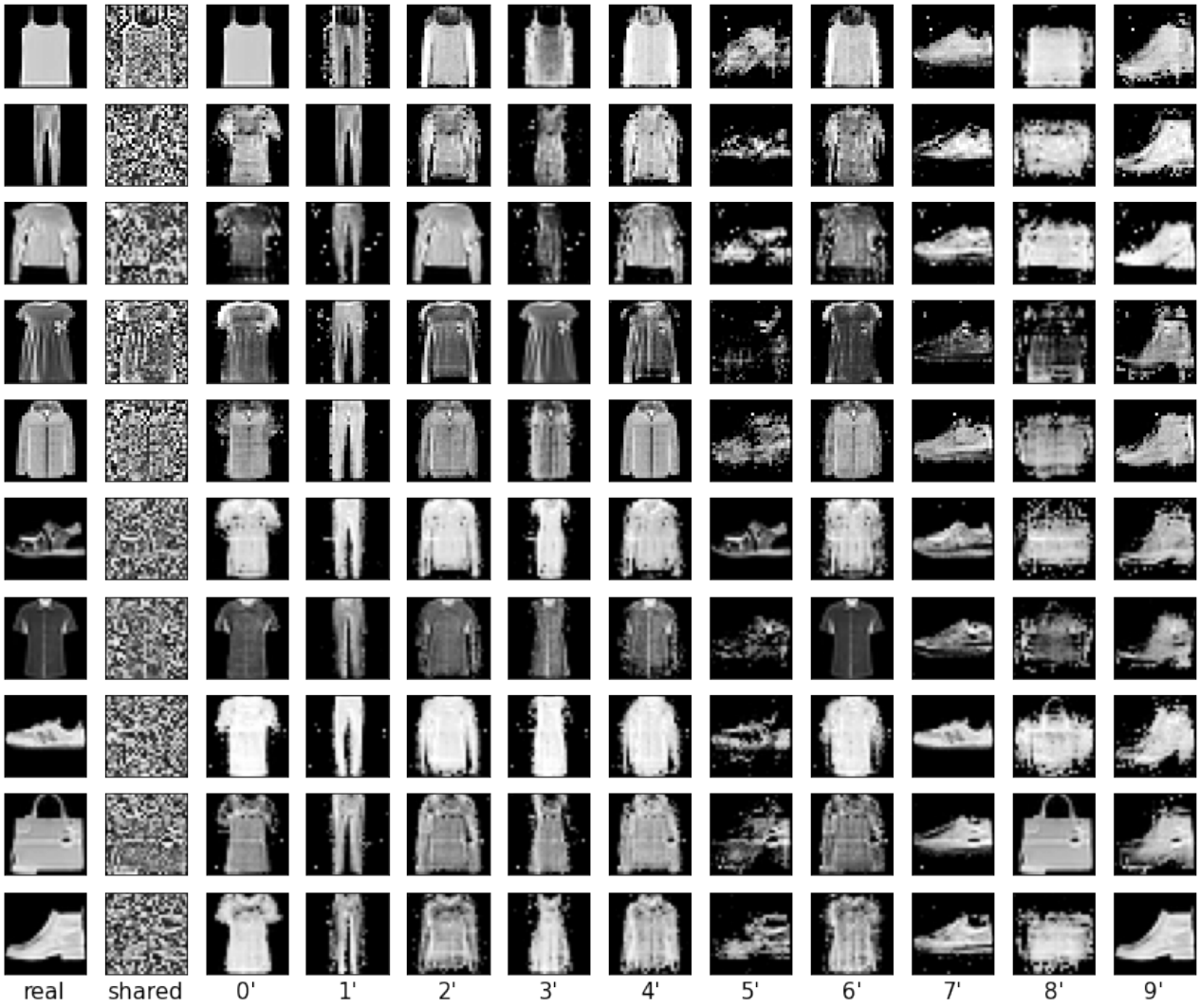


Figure 16: Multi-distribution ($M = 10$) results for FashionMNIST with DD.

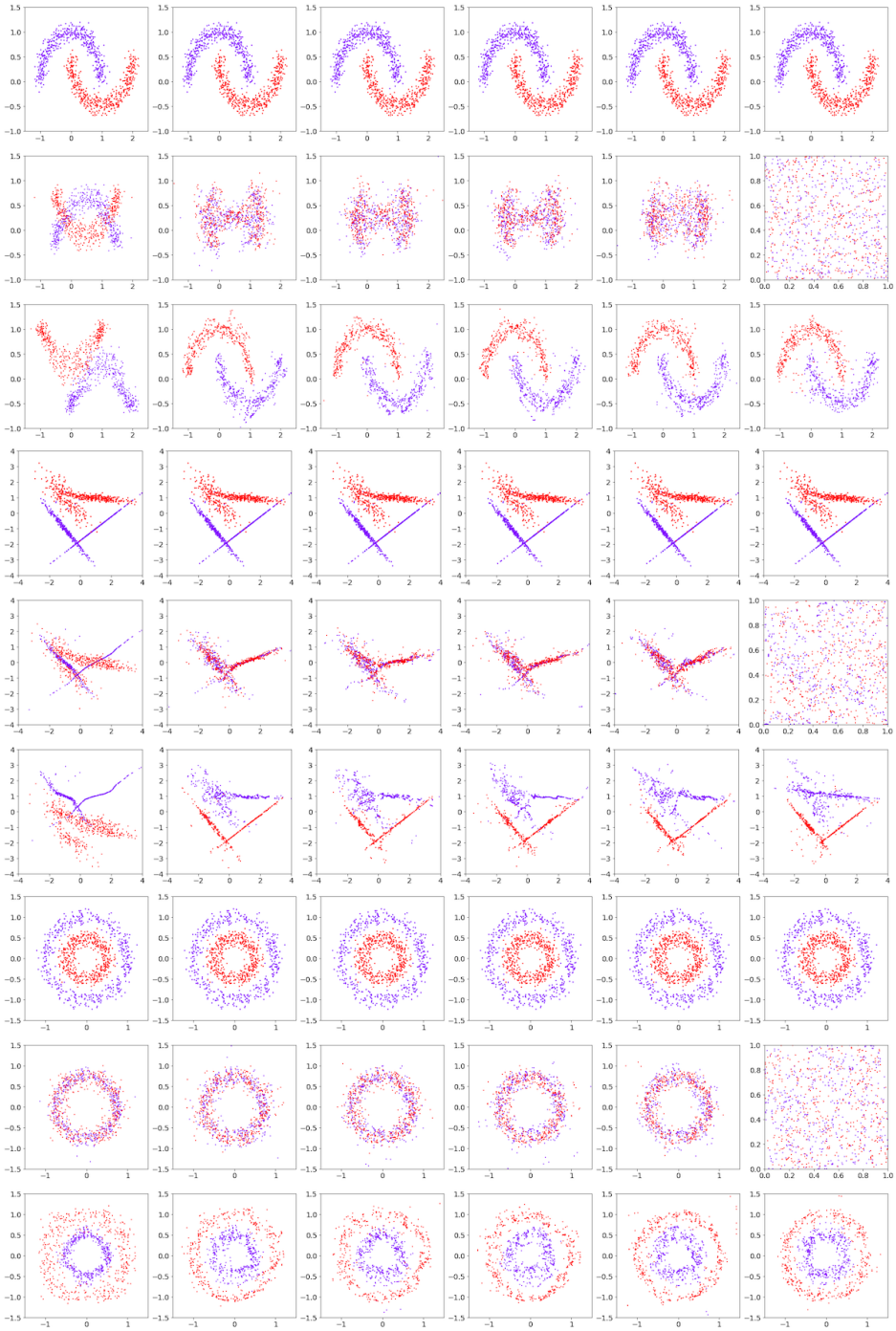


Figure 17: Expanded figure of 2D Data ($M = 2$). In each sub figure, the first row represents the original data. The second row represents the latent distribution. The third row represents the flipped distribution. The columns from left to right represent the model: NB, INB, NB-INB, Rand-NB, NB-Rand-NB, DD.

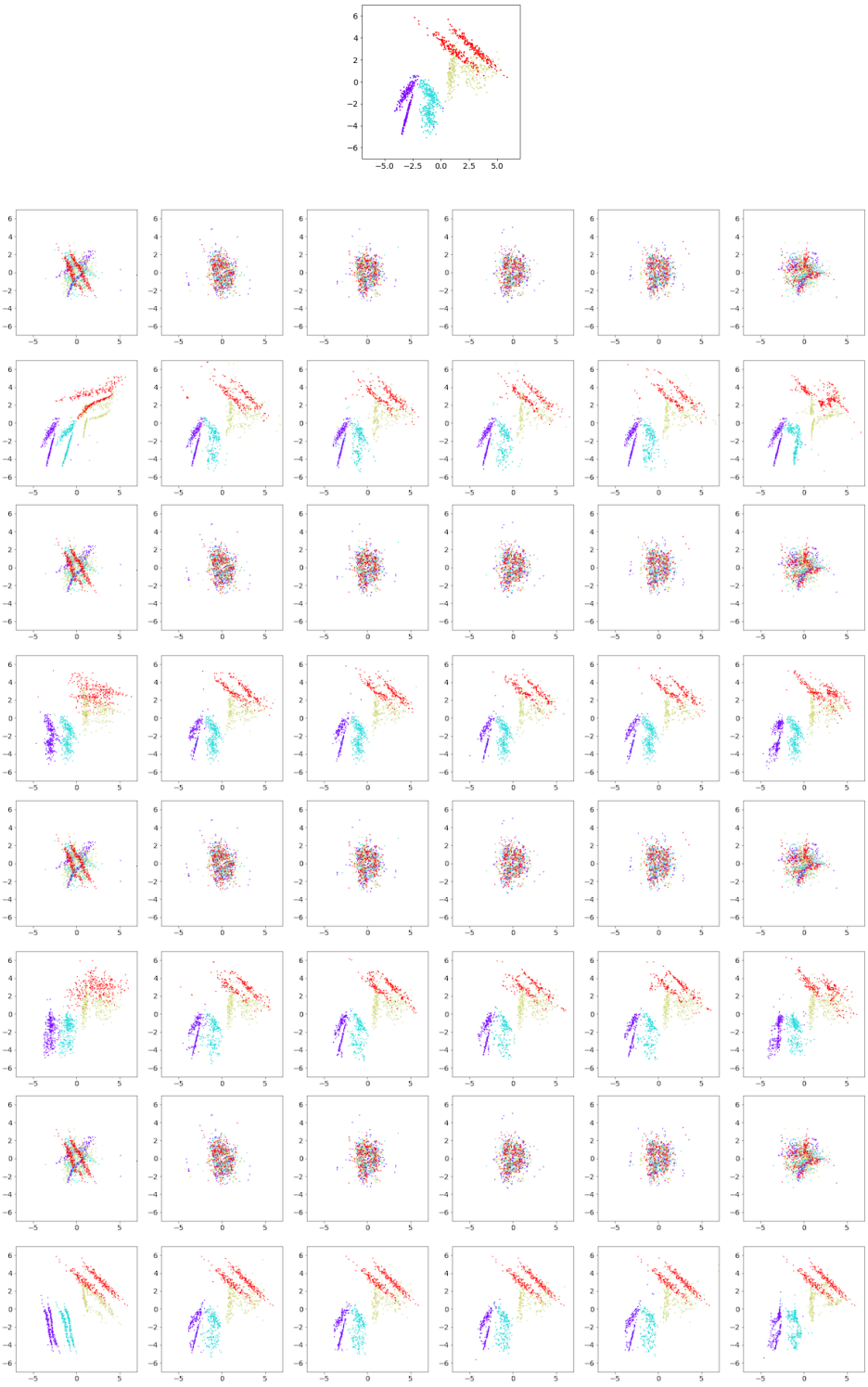


Figure 18: Expanded figure of 2D Random Pattern ($M = 4$). The columns from left to right represent the model: NB, INB, NB-INB, Rand-NB, NB-Rand-NB, DD. The top image is the original distribution. Each pair of rows represents the translation of samples from one class distribution to all other class distributions. We can translate every class distribution to every other class distribution since all functions are invertible. The pairs of rows are the results of translating from different source distributions, i.e., class 1 (purple), class 2 (turqoise), class 3 (yellow), and class 4 (red) distributions respectively. The top of each pair is the shared latent representation (the same across all rows) whereas the bottom row shows the generated data. Note that if the source and target distribution are the same, e.g., from class 1 to class 1, the output distribution will be exactly as in the original since our transformations are invertible.

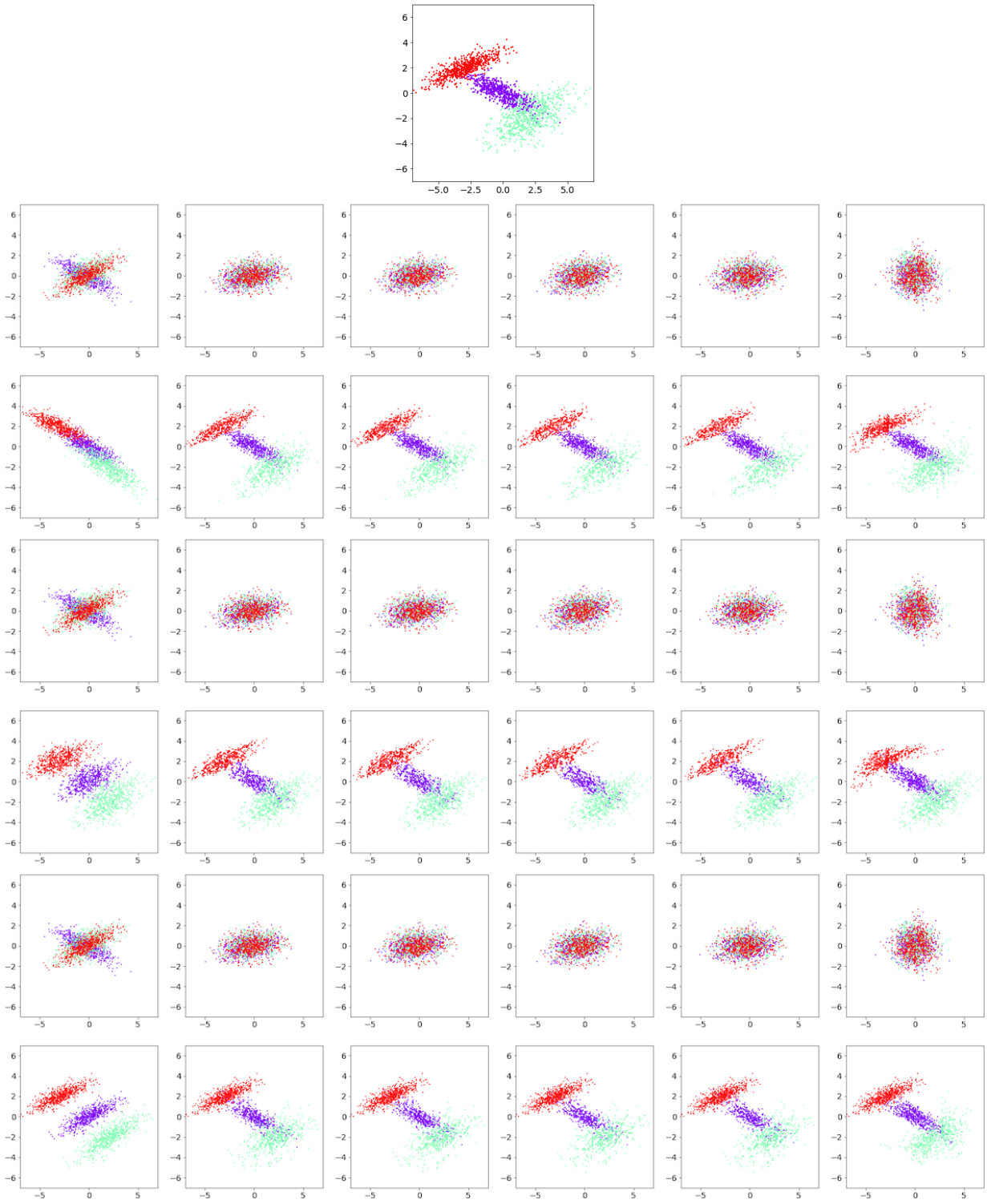


Figure 19: Expanded figure of 2D Gaussian ($M = 3$). The columns from left to right represent the model: NB, INB, NB-INB, Rand-NB, NB-Rand-NB, DD. The top image is the original distribution. See caption of Fig. 18 for explanation of each pair of rows.