Cooperative Distribution Alignment via JSD Upper Bound

Anonymous Author(s)
Affiliation
Address
email

Abstract

Unsupervised distribution alignment estimates a transformation that maps two or more source distributions to a shared aligned distribution given only samples from each distribution. This task has many applications including generative modeling, unsupervised domain adaptation, and socially aware learning. Most prior works use adversarial learning (i.e., min-max optimization), which can be challenging to optimize and evaluate. A few recent works explore non-adversarial flow-based (i.e., invertible) approaches, but they lack a unified perspective and are limited in efficiently aligning multiple distributions. Therefore, we propose to unify and generalize previous flow-based approaches under a single non-adversarial framework, which we prove is equivalent to minimizing an upper bound on the Jensen-Shannon Divergence (JSD). Importantly, our problem reduces to a min-min, i.e., cooperative, problem and can provide a natural evaluation metric for unsupervised distribution alignment. We present empirical results of our framework on both simulated and real-world datasets to demonstrate the benefits of our approach.

1 Introduction

In many cases, a practitioner has access to multiple related but distinct distributions such as agricultural measurements from two farms, experimental data collected in different months, or sales data before and after a major event. *Unsupervised* distribution alignment (UDA) is the ML task aimed at aligning these related but distinct distributions in a shared space, *without* any pairing information between the samples from these distrbibutions (i.e., unsupervised). This task has many applications such as generative modeling (e.g., [1]), unsupervised domain adaptation (e.g., [2, 3]), batch effect mitigation in biology (e.g., [4]), and fairness-aware learning (e.g., [5]).

The most common approach for obtaining such alignment transformations stems from Generative Adversarial Networks (GAN)[6], which can be viewed as minimizing a *lower bound* on the Jensen-Shannon Divergence (JSD) between real and generated distributions. The lower bound is tight if and only if the inner maximization is solved perfectly. CycleGAN [1] maps between *two* datasets via two GAN objectives between the two domains and a cycle consistency loss, which encourages approximate invertibility of the transformations.

However, adversarial learning can be challenging to optimize in practice (see e.g. [7, 8, 9, 10, 11]) in part because of the competitive nature of the min-max optimization problem. Perhaps more importantly, the research community has resorted to surrogate evaluation metrics for GAN because likelihood computation is intractable. Specifically, the commonly accepted Frechet Inception Distance (FID) [12] is only applicable to image or auditory data, where publicly available classifiers trained on large-scale data exist. Moreover, the concrete implementation of FID can have issues due to seemingly trivial changes in image resizing algorithms [13].

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Recently, flow-based methods with a tractable likelihood have been proposed for the UDA task [2, 14]. AlignFlow [2] leverages invertible models to make the model cycle-consistent (i.e., invertible) *by construction* and introduces exact log-likelihood loss terms derived from standard flow-based generative models as complementary loss terms to the adversarial loss terms. On the other hand, log-likelihood ratio minimizing flows (LRMF) [14] use invertible flow models and density estimation for distribution alignment without adversarial learning and define a new metric based on the log-likelihood ratio.

However, AlignFlow lacks an explicit alignment metric and assumes that the shared density model for all distributions is a fixed Gaussian. Also, the LRMF metric is limited because it is only defined for two distributions and strongly depends on the shared density model class; if the target distribution is not in this model class, LRMF can only partially align distributions.

To address these issues, we unify existing flow-based methods (both AlignFlow and LRMF) under a common cooperative (i.e., non-adversarial) framework by proving that a minimization over a shared density model is a variational upper bound of the JSD. The unifying theory also suggests a natural domain-agnostic metric for UDA that can be applied to any domain including tabular data (where FID is inapplicable). This metric is analogous to the Evidence Lower Bound (ELBO), i.e., it is a variational bound that is useful for both training models and comparing models via held-out test evaluation. Furthermore, this unification enables straightforward and parameter-efficient multi-distribution alignment because the distributions share a latent space density model. We summarize our contributions as follows:

- We prove that a minimization over a shared variational density model is a *variational upper bound* on a generalized version of JSD that allows for more than two distributions. Importantly, we theoretically quantify the bound gap and show that it can be made tight if the density model class is flexible enough.
- Based on this JSD upper bound, we derive a novel unified framework for cooperative (i.e., non-adversarial) flow-based UDA (including a novel domain-agnostic AUB metric) and explain its relationship to prior flow-based alignment methods.
- Throughout experiments, we demonstrate that our framework consistently shows superior performance in both our proposed and existing measures on simulated and real-world datasets. We also quantitatively verify that our model is more parameter-efficient than the baseline models.

Notation We will denote distributions as $P_X(\boldsymbol{x})$ where X is the corresponding random variable. Invertible functions will be denoted by $T(\cdot)$. We will use $X_j \sim P_{X_j}$ to denote the observed random variable from the j-th distribution. We will use $Z_j \triangleq T_j(X_j) \sim P_{Z_j} \equiv P_{T_j(X_j)}$ to denote the latent random variable of the j-th distribution after applying T_j to X_j (and note that $X_j = T_j^{-1}(Z_j)$). We will denote the mixtures of these observed or latent distributions as $P_{X_{\text{mix}}} \triangleq \sum_j w_j P_{X_j}$ and $P_{Z_{\text{mix}}} \triangleq \sum_j w_j P_{Z_j}$, where $\boldsymbol{w}$ is a probability vector. We denote KL divergence, entropy, and cross entropy as $\mathrm{KL}(\cdot,\cdot)$, $\mathrm{H}(\cdot)$, and $\mathrm{H_c}(\cdot,\cdot)$, respectively, where $\mathrm{KL}(P,Q) = \mathrm{H_c}(P,Q) - \mathrm{H}(P)$.

2 Alignment Upper Bound Loss

In this section, we will introduce our main theoretical result proving an upper bound on the generalized JSD divergence, deriving our loss function based on this, and then showing that minimizing this upper bound results in aligned distributions assuming large enough capacity of the model components.

Background: Normalizing Flows and Invertible Models Normalizing flows are generative models that have tractable distributions where exact density evaluations and efficient samplings are ensured [15]. Such models leverage the change of variables formula to create an invertible mapping T such that $P_X(\boldsymbol{x}) = P_Z(T(\boldsymbol{x}))|J_T(\boldsymbol{x})|$ where P_Z is a known latent distribution and $|J_T(\boldsymbol{x})|$ is the absolute value of determinant of the Jacobian of the invertible map T. For sampling in distribution P_X, one need to first sample from the latent distribution P_Z and then apply the inverse transform T^{-1}. Therefore the key challenge in designing invertible models is to have computationally efficient inverse evaluation and Jacobian determinant calculation. Many approaches have been

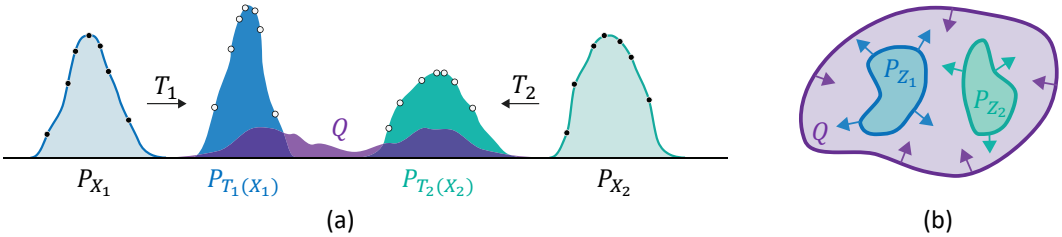


Figure 1: Overview of our proposed method. In our framework, domain specific transformation functions $\{T_j(\cdot)\}_{j=1}^k$ and a density model Q are cooperatively trained to make the transformed representations be indistinguishable in the shared latent space. **(a)** 1-D example. By minimizing our proposed AUB loss, the transformation functions T_1 and T_2 are trained to map the corresponding distributions P_{X_1} and P_{X_2} to latent distributions $P_{T_1(X_1)}$ and $P_{T_2(X_2)}$ that have higher likelihood with respect to a base distribution Q. The density model Q, on the other hand, is trained to fit the mixture of the latent distributions $P_{T_1(X_1)}$ and $P_{T_2(x_2)}$. **(b)** Intuitively, the optimization process of our method can be seen to make the Q distribution tight around the mixture of latent distributions to increase the likelihood (i.e., MLE) while the transformation functions T_1 and T_2 are encouraged to expand to fill the latent space defined by Q. Eventually, the latent distributions and Q are converge to be the same distribution, which means that they are aligned.

proposed by parameterizing mapping function T as deep neural networks including autoregressive structures [16, 17], coupling layers [18][19], oridinary differential equations[20], and invertible residual networks[21, 22]. Flow models can be then learned efficiently by maximizing the likelihood for the given data.

Background: Generalized JSD We remind the reader of the generalized Jensen-Shannon divergence for more than two distributions, where the standard JSD is recovered if $w_1 = w_2 = 0.5$.

Definition 2.1 (Generalized Jensen-Shannon Divergence (GJSD) [23])**.** Given k distributions $\{P_{X_j}\}_{j=1}^k$ and a corresponding probability weight vector $\boldsymbol{w}$, the generalized Jensen-Shannon divergence is defined as (proof of equivalence in supplementary material.):

$$\begin{aligned}\mathrm{GJSD}_{\boldsymbol{w}}(P_{X_1}, \cdots, P_{X_k}) &\triangleq \textstyle\sum_j w_j \, \mathrm{KL}(P_{X_j}, \sum_j w_j P_{X_j}) \\ &\equiv \mathrm{H}\left(\textstyle\sum_j w_j P_{X_j}\right) - \textstyle\sum_j w_j \, \mathrm{H}(P_{X_j}) \, .\end{aligned}$$

2.1 GJSD Variational Upper Bound

The goal of distribution alignment is to find a set of transformations $\{T_j(\cdot)\}_{j=1}^k$ (which will be invertible in our case) such that the latent distributions align, i.e., $P_{T_j(X_j)} = P_{T_{j'}(X_{j'})}$ or equivalently $P_{Z_j} = P_{Z_{j'}}$ for all $j \neq j'$. Given the properties of divergences, this alignment will happen if and only if $\mathrm{GJSD}(P_{Z_1}, \cdots, P_{Z_k}) = 0$. Thus, ideally, we would minimize GJSD directly, i.e.,

$$\min_{T_1, \cdots, T_k \in \mathcal{T}} \mathrm{GJSD}(P_{T_1(X_1)}, \cdots, P_{T_k(X_k)}) \equiv \min_{T_1, \cdots, T_k \in \mathcal{T}} \mathrm{H}\left(\textstyle\sum_j w_j P_{T_j(X_j)}\right) - \textstyle\sum_j w_j \, \mathrm{H}(P_{T_j(X_j)}) \, , \tag{1}$$

where $\mathcal{T}$ is a class of invertible functions. However, we cannot evaluate the entropy terms in Eqn. 1 because we do not know the density of P_{X_j}; we only have samples from P_{X_j}. Therefore, we will upper bound the first entropy term in Eqn. 1 ($\mathrm{H}(\sum_j w_j P_{T_j(X_j)})$) using a variational density model and decompose the other entropy terms by leveraging the change of variables formula for invertible functions.

Theorem 2.2 (GJSD Variational Upper Bound)**.** *Given an variational density model class $\mathcal{Q}$, we form a GJSD variational upper bound:*

$$\mathrm{GJSD}_{\boldsymbol{w}}(P_{Z_1}, \cdots, P_{Z_k}) \leq \min_{Q \in \mathcal{Q}} \mathrm{H_c}(P_{Z_{\text{mix}}}, Q) - \textstyle\sum_j w_j \, \mathrm{H}(P_{Z_j}) \, ,$$

where the bound gap is exactly $\min_{Q \in \mathcal{Q}} \mathrm{KL}(P_{Z_{\text{mix}}}, Q)$.

Proof of Theorem 2.2. For any $Q \in \mathcal{Q}$, we have the following upper bound:

$$\begin{aligned}\mathrm{GJSD}_{\boldsymbol{w}}(P_{Z_1}, \cdots, P_{Z_k}) &= \underbrace{\mathrm{H_c}(P_{Z_{\text{mix}}}, Q) - \mathrm{H_c}(P_{Z_{\text{mix}}}, Q)}_{=0} + \mathrm{H}(P_{Z_{\text{mix}}}) - \textstyle\sum_j w_j \, \mathrm{H}(P_{Z_j}) \\ &= \mathrm{H_c}(P_{Z_{\text{mix}}}, Q) - \mathrm{KL}(P_{Z_{\text{mix}}}, Q) - \textstyle\sum_j w_j \, \mathrm{H}(P_{Z_j}) \\ &\leq \mathrm{H_c}(P_{Z_{\text{mix}}}, Q) - \textstyle\sum_j w_j \, \mathrm{H}(P_{Z_j}) \,,\end{aligned}$$

where the first equals is merely inflating by $\mathrm{H_c}(P_{Z_{\text{mix}}}, Q)$, the inequality is by the fact that KL divergence is non-negative and the bound gap is equal to $\mathrm{KL}(P_{Z_{\text{mix}}}, Q)$. The Q that achieves the minimum in the upper bound is equivalent to the Q that minimizes the bound gap, i.e., $Q^* = \arg\min_{Q \in \mathcal{Q}} \mathrm{H_c}(P_{Z_{\text{mix}}}, Q) - \sum_j w_j \, \mathrm{H}(P_{Z_j}) = \arg\min_{Q \in \mathcal{Q}} \mathrm{H_c}(P_{Z_{\text{mix}}}, Q) - \mathrm{H}(P_{Z_{\text{mix}}}) = \arg\min_{Q \in \mathcal{Q}} \mathrm{KL}(P_{Z_{\text{mix}}}, Q)$, where the second equality is because the entropy terms are constant with respect to Q and the last is by the definition of KL divergence. □

The tightness of the bound depends on how well the class of density models $\mathcal{Q}$ (e.g., mixture models, normalizing flows, or autoregressive densities) can approximate $P_{Z_{\text{mix}}}$; notably, the bound can be made tight if $P_{Z_{\text{mix}}} \in \mathcal{Q}$. Also, one key feature of this upper bound is that the cross entropy term can be evaluated using only samples from P_{X_j} and the transformations T_j, i.e., $\mathrm{H_c}(P_{Z_{\text{mix}}}, Q) = \sum_j w_j \mathbb{E}_{P_{X_j}}[-\log Q(T_j(\boldsymbol{x}_j))]$. However, we still cannot evaluate the other entropy terms $\mathrm{H}(P_{Z_j})$ since we do not know the densities of P_{Z_j} (or P_{X_j}). Thus, we leverage the fact that the T_j functions are invertible to define an entropy change of variables.

Lemma 2.3 (Entropy Change of Variables)**.** *Let $X \sim P_X$ and $Z \triangleq T(X) \sim P_Z$, where T is an invertible transformation. The entropy of Z can be decomposed as follows:*

$$\mathrm{H}(P_Z) = \mathrm{H}(P_X) + \mathbb{E}_{P_X}[\log |J_T(\boldsymbol{x})|] \,, \tag{2}$$

where $|J_T(\boldsymbol{x})|$ is the determinant of the Jacobian of T.

The key insight from this lemma is that $\mathrm{H}(P_X)$ is a constant with respect to T and can thus be ignored when optimizing T, while $\mathbb{E}_{P_X}[\log |J_T(\boldsymbol{x})|]$ can be approximated using only samples from P_X (formal proof in supplementary material).

2.2 Alignment Upper Bound (AUB)

Combining Theorem 2.2 and Lemma 2.3, we can arrive at our final objective function which is equivalent to minimizing the variational upper bound on the GJSD:

$$\mathrm{GJSD}_{\boldsymbol{w}}(P_{Z_1}, \cdots, P_{Z_k}) \leq \min_{Q \in \mathcal{Q}} \mathrm{H_c}(P_{Z_{\text{mix}}}, Q) - \textstyle\sum_j w_j \, \mathrm{H}(P_{Z_j}) \tag{3}$$

$$= \min_{Q \in \mathcal{Q}} \textstyle\sum_j w_j \mathbb{E}_{P_{X_j}}[-\log Q(T_j(\boldsymbol{x})) |J_{T_j}(\boldsymbol{x})|] - \textstyle\sum_j w_j \, \mathrm{H}(P_{X_j}) \,, \tag{4}$$

where the last term $-\sum_j w_j \, \mathrm{H}(P_{X_j})$ is constant with respect to T_j functions so they can be ignored. We formally define this loss function as follows.

Definition 2.4 (Alignment Upper Bound Loss)**.** Given k continuous distributions $\{P_{X_j}\}_{j=1}^k$, a class of continuous distributions $\mathcal{Q}$, and a probability weight vector $\boldsymbol{w}$, the alignment upper bound loss is defined as follows:

$$\mathcal{L}_{\mathrm{AUB}}(T_1, \cdots, T_k; \{P_{X_j}\}_{j=1}^k, \mathcal{Q}, \boldsymbol{w}) \triangleq \min_{Q \in \mathcal{Q}} \textstyle\sum_j w_j \mathbb{E}_{P_{X_j}}[-\log |J_{T_j}(\boldsymbol{x})| \, Q(T_j(\boldsymbol{x}))] \,, \tag{5}$$

where T_j are invertible and $|J_{T_j}(\boldsymbol{x})|$ is the absolute value of the Jacobian determinant.

Notice that this alignment loss can be seen as learning the best base distribution given fixed flow models T_j. We now consider the theoretical optimum if we optimize over all invertible functions.

Theorem 2.5 (Alignment at Global Minimum of $\mathcal{L}_{\mathrm{AUB}}$)**.** *If $\mathcal{L}_{\mathrm{AUB}}$ is minimized over the class of all invertible functions, a global minimum of $\mathcal{L}_{\mathrm{AUB}}$ implies that the latent distributions are aligned, i.e., $P_{T_j(X_j)} = P_{T_{j'}(X_{j'})}$ for all $j \neq j'$. Notably, this result holds regardless of $\mathcal{Q}$.*

Informally, this can be proved by showing that the problem decouples into separate normalizing flow losses where Q is the base distribution and the optimum is achieved only if $P_{T_j(X_j)} = Q$ for all T_j (formal proof in supplementary material). This alignment of the latent distributions also implies the translation between any of the *observed* component distributions. The proof follows directly from Theorem 2.5 and the change of variables formula.

Corollary 2.6 (Translation at Global Minimum of $\mathcal{L}_{\text{AUB}}$)**.** *Similar to Theorem 2.5, a global minimum of $\mathcal{L}_{\text{AUB}}$ implies translation between any component distributions using the inverses of T_j, i.e., $P_{T_{j'}^{-1}(T_j(X_j))} = P_{X_{j'}}$ for all $j \neq j'$.*

As seen in Alg. 1, we use a simple alternating optimization scheme for training our translation models and variational distribution with cooperative (i.e., min-min) AUB objective, i.e., we aim to optimize:

$$\min_{T_1,\cdots,T_k \in \mathcal{T}} \min_{Q \in \mathcal{Q}} \sum_j w_j \mathbb{E}_{P_{X_j}} \left[-\log |J_{T_j}(\boldsymbol{x})| \, Q(T_j(\boldsymbol{x})) \right]. \tag{6}$$

We emphasize that our framework allows *any* invertible function for T_j (e.g., coupling-based flows [18], neural ODE flows [20], or residual flows [21]) and *any* density model class for Q_z (e.g., kernel densities (in low dimensions), mixture models, autoregressive densities [24], or normalizing flows [19]). Even VAEs [25] could be used where the log likelihood term is upper bounded by the negative ELBO, which will ensure the objective is still an upper bound of GJSD.

Algorithm 1 Training algorithm for our model

Input: Datasets $\{X_j\}_{j=1}^k$ for k domains; n_j as number of data points in domain j; normalizing flow models $\{T_j(x_j;\theta_j)\}_{j=1}^k$; density model $Q(z;\phi)$; learning rate η; maximum epoch E_{max}
Output: $\{\hat{\theta}\}_{j=1}^k$;
for epoch $= 1, E_{max}$ **do**
  for each batch $\{x_j\}_{j=1}^k$ **do**
    $\phi \leftarrow \phi + \eta \nabla_\phi \frac{1}{k} \sum_{j=1}^k \frac{1}{n_j} \sum_{i=1}^{n_j} \log Q(T_j(x_{i,j};\theta_j);\phi)$
  end for
  for each batch $\{x_j\}_{j=1}^k$ **do**
    $\forall j, \quad \theta_j \leftarrow \theta_j + \eta \nabla_{\theta_j} \frac{1}{n_j} \sum_{i=1}^{n_j} \log |J_{T_j}(x_{i,j};\theta_j)| \, Q(T_j(x_{i,j};\theta_j);\phi)$
  end for
end for

AUB for UDA is like ELBO for density estimation Although AUB and ELBO are for fundamentally different tasks, we would like to point out the similarities between AUB and the ELBO. First, both are *variational* bounds of the quantity of interest where the tightness of the bounds depend on the optimization of the variational distributions. Second, both can be made tight if the class of variational distributions is powerful enough. Third, both can be used to train a model by minimizing the objective on training data. Fourth, while neither can be used as an absolute performance metric, they can both be used to evaluate the *relative* performance of models on held-out test data. Thus, we believe AUB can be used for UDA like ELBO has been used for density estimation with the same strengths (and weaknesses such as it being a relative metric).

3 Relationship to Prior Works

AlignFlow without adversarial terms is a special case AlignFlow [2] *without* adversarial loss terms is a special case of our method for two distributions where the density model class $\mathcal{Q}$ only contains the standard normal distribution (i.e., a singleton class). Thus, AlignFlow can be viewed as initially optimizing a poor upper bound on JSD; however, the JSD bound becomes tighter as training progresses because the latent distributions *independently* move towards the same normal distribution.

LRMF is special case with only one transformation Log-likelihood ratio minimizing flows (LRMF) [14] is also a special case of our method for only two distributions, where one transformation is fixed at the identity (i.e., $T_2 = \text{Id}$). While the final LRMF objective is a special case of ours, the theory is developed from a different but complementary perspective. The LRMF metric depends on the shared density model class, which enables a zero point (or absolute value) of the metric to be estimated but requires fitting extra density models. [14] do not uncover the connection of the objective as an upper bound on JSD regardless of the density model class. Additionally, to ensure alignment, LRMF requires that the density model class includes the true target distribution because only one invertible transform is used, while our approach can theoretically align even if the shared density model class is weak (see Theorem 2.5 and our simulated experiments).

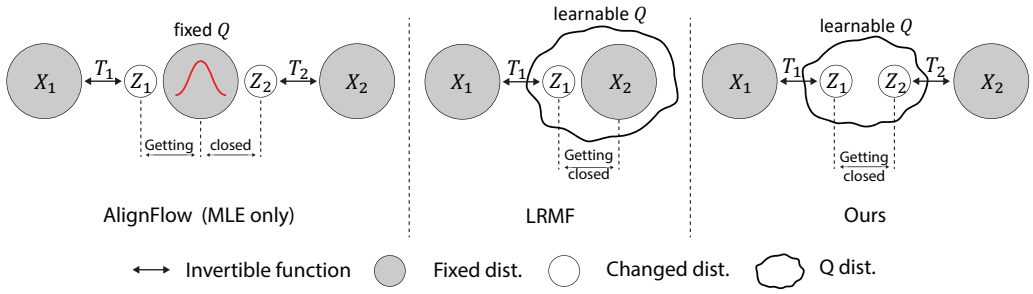
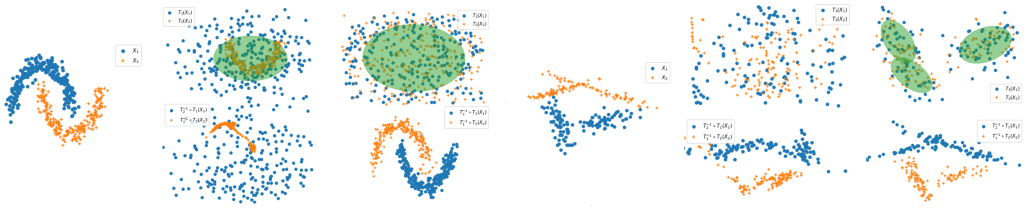


Figure 2: High-level comparison with the baseline models. AlignFlow, LRMF, and our setup are illustrated in a row from left. Transformation functions in AlignFlow are independently trained to be fitted to the fixed standard gaussian distribution. T_1 in LRMF is aimed to directly map the given X_1 to another image distribution X_2. The density model Q in LRMF is not fixed and learned to fit to $\{Z_1 \cap X_2\}$. In our setup, T_1 and T_2 are trained to obtain the high likelihood from the learnable Q distribution which is fitted to the shared latent distributions. In every setup, the latent distributions Z_1 and Z_2 are getting closer to the target distribution as training goes by. Details are provided in section 3.



(a) Original (b) LRMF (c) AUB(ours) (d) Original (e) AlignFlow (f) AUB(ours)

Figure 3: Top row is latent space and bottom is the data translated into the other space. (a-c) LRMF, which only has *one* transformation T may not be able to align the datasets if the density model class $\mathcal{Q}$ is not expressive enough (in this case Gaussian distributions) while using two transformations as in our framework can align them. (d-f) AlignFlow (without adversarial terms) may not align because Q_z is fixed at a standard normal, while our approach with learnable mixture of Gaussians for Q_z is able to learn an alignment (both use the same T_j models).

4 Experiment

We analyze the performance of our proposed framework comparing to the relevant flow-based baseline models. In subsection 4.1, we conduct our experiment in toy dataset to clearly show the benefit of our method over the baseline flow-based models in a controlled environment (implementation details are in supplemantary material). In subsection 4.2 and subsection 4.3, we demonstrate our superiority over baseline flow-based models on real-world datasets including tabular data and high-dimensional MNIST data.

4.1 Toy dataset comparison with related works

Single T vs. Double T (LRMF vs. Ours) We first compare our method with LRMF [14] method. We construct the experiment to have the task: Translation between the two half-circled distributions: X_1 and X_2 in the moons dataset. In this example, we made two models, one with LRMF setup and one with our AUB setup. As illustrated in Fig. 3, the LRMF method fails to transform between X_1 and X_2. Even though Q can model well enough for $T_1(X_1)$, Q can only model the mean and variance of X_2 which is obviously not informative enough. Therefore, the LRMF fails to transform between two datasets. While in the AUB setup, both $T_1(X_1)$ and $T_2(X_2)$ are modeled to the same distribution which Q can be learned to fit with high likelihood which leads to better translation results. In conclusion, the performance of the LRMF model is limited by the power of the density model Q which means if Q fails to model one of the domain distribution in high likelihood, data alignment cannot be achieved with good performances.

Simple Fixed Q vs. Learnable Q (AlignFlow vs. Ours) Next we compare our method with AlignFlow[2, 3] setup. We construct the experiment to have the task: Transform between the two random patterns X_1 and X_2 from the randomly generated datasets. Again, we made two models with AlignFlow and our AUB setups respectively. As illustrated in Fig. 3, the AlignFlow method fails to transform between X_1 and X_2, because the transformed dataset $T_1(X_1)$ and $T_2(X_2)$ failed to reach the normal distribution Q. While in the AUB setup, the density model Q is learned to help fit the transformed distributions $T_1(X_1)$ and $T_2(X_2)$, which allows them to be aligned with each other easier. In conclusion, the performance of the AlignFlow model is limited by the performances of the invertible functions.

4.2 Unsupervised Domain Alignment on Tabular Datasets

To demonstrate the benefits of our framework, we conduct two experiments on real-world tabular datasets. In both experiments, we used four UCI tabular datasets [26] (MINIBOONE, GAS, HEPMASS, and POWER), following the same preprocessing as the MAF paper [27]. Train, validation, and test sets are 80%, 10%, and 10% of the data respectively. Also, the experiments are measured by test AUB defined in Definition 2.4, where a lower AUB score indicates better performance (see end of section 2.2 for discussion on using test AUB for evaluation). We emphasize that there is no natural metric for evaluating GAN-based alignment methods on tabular datasets. We believe this demonstrates one of the key benefits of our proposed framework over a GAN-based approach.

Our first experiment was designed to compare alignment performance between our proposed method and the baseline models. To separate each dataset into two distributions, we choose the last input feature from each dataset and discretize it based on whether it is higher or lower than the median value, which ensures the datasets are of equal size. Given the divided dataset, two transformation functions (T) are trained to accomplish distribution alignment. Regarding baseline models, we use AlignFlow with MLE, Adv. only, and hybrid versions and LRMF on top of the original implementation. Because AlignFlow Adv. and hybrid setups optimize over a mixed objective of AUB (special case) and adversarial losses, to be fair, we additionally fit an identity-initialized Q to the final T's. We also use the same T and Q functions (if exists in setup) with ours across all the baselines. Note that the Q is from the same class of distributions $\mathcal{Q}$ with our approach.

As shown in Table 1, our method shows better performance compared to other methods across all datasets. In particular, because ours and LRMF (trained with learnable Q) outperform AlignFlow MLE (trained with fixed Q), we can see that learnable Q plays an important role in distribution alignment. We also observe that ours shows better performance than LRMF, where the gap may come from aligning in the shared latent space (ours) rather than aligning in the original data space (LRMF). AlignFlow Adv. only and hybrid versions show worse performance than ours with a large gap, which implies our proposed cooperative training is competitive to adversarial methods in aligning distributions of tabular data.

Table 1: AUB score for the domain alignment task in four tabular datasets. AUB score is shown in nats (the lower the better). Numbers below each dataset indicate the number of features for the dataset.

	MINIBOONE (42)	GAS (7)	HEPMASS (20)	POWER (5)
LRMF	12.79	-6.17	18.49	-0.93
AF (MLE)	14.08	-6.52	19.37	-0.77
AF (Adv. only)	18.18	-3.15	21.70	-0.39
AF (hybrid)	19.49	-3.76	21.42	-0.43
Ours	**12.11**	**-7.09**	**18.26**	**-1.19**

Table 2: Parameter efficiency of our proposed framework. Unit for the number of parameters is million.

	T param.	Q param.	total	AUB
AF (MLE) T: RealNVP (5)	1.46	0	1.46	20.16
AF (MLE) T: RealNVP (10)	4.54	0	4.54	19.85
Ours T: R.(5), Q: R.(10)	1.46	0.57	**2.03**	**18.82**

The second experiment is designed to show that our proposed idea is more efficient in terms of model parameter compared to the baseline model. For implementing experiments with multiple distributions to align rather than just two, we separate each dataset into 8 domains by choosing last three features from a given dataset and divide the dataset by the three medians, e.g., $(+++)$,$(++-)$, ..., $(---)$. The results are shown in Table 2 RealNVP (5) and RealNVP (10) indicates the number of coupling layers used in the RealNVP architecture respectively. LRMF comparison is excluded because it is designed for aligning only two distributions.

From a comparison between two AlignFlow setups with RealNVP (5) and RealNVP (10), we can see that AlignFlow can achieve better alignment with more than double amount of parameters. However, as can be observed in a comparison between AlignFlow with RealNVP (10) and ours, our proposed method can achieve similar performance with the baseline model but with less than half parameters. These two results suggest that our model can be more parameter efficient than AlignFlow for multi-distribution alignment. We hypothesize that our approach can scale better to any number of distributions because our Q shares parameters across the 8 distributions and can capture the similarities between distributions.

4.3 Unsupervised Distribution Alignment on MNIST Dataset

We perform an image translation task on MNIST dataset[1] [28] to demonstrate that our distribution alignment method could be applied to high-dimensional datasets. We train ours and baseline models with the digit images of 0, 1, and 2, and compare the translated results both quantitatively and qualitatively. Specifically, We use RealNVP invertible models for all translation maps T_j, as well as the density model Q. Note that all methods are flow-based models and thus images translated back to original domain are exactly the same, which implies exact cycle consistency. Details for the setup of the experiments are available in supplementary material.

As represented in Fig. 4, both of our approaches outperform the baseline models in terms of FID and AUB. The lower score in AUB indicates our model has the tighter bound than baseline model. This implies the statistical distance between the shared distribution $\{Z_j\}_{j=1}^k$ and our density model Q is smaller than baselines, meaning our model has the better distribution alignment performance. We believe this result comes from our model setup, i.e., a learnable shared density model and transformation to a shared latent space. Specifically, AlignFlow with a fixed standard normal distribution as their Q obtains worse AUB because the Q is not powerful enough to model the complex shared space trained from the real world dataset. On the other hand, LRMF shows the lack of stability when trained with the relatively simple models that we are using across all methods, i.e., RealNVP T and RealNVP Q. We expect this is caused by the restrictions of only using one T for translation without a shared latent space and the fact that the Q distribution must be able to model the target distribution to ensure alignment.

The better quantitative performance of our methods can be corroborated by the qualitative results as seen in Fig. 4. AlignFlow shows less stable translation results than our method especially for translating to digit '2' from digits '0' and '1'. We believe this phenomenon comes from the lack of expressivity of their Q model. On the other hand, our model shows stable results across all translation cases, which are also quantitatively verified via the lower FID score. Note that we have a shared space, so different transformation functions for each domain are trained together. This takes advantages over the baseline in terms of sample complexity and computational complexity compared to multiple independent flows as in AlignFlow.

	FID				AUB			
	0↔1	0↔2	1↔2	Avg.	0↔1	0↔2	1↔2	Avg.
LRMF	224.02	141.70	182.31	182.68	-713	-592	-1323	-876
AF (MLE)	38.90	71.17	61.33	57.13	-4797	-4504	-4834	-4711
Ours	**31.26**	**43.29**	**41.55**	**38.70**	**-4824**	**-4555**	**-4862**	**-4747**

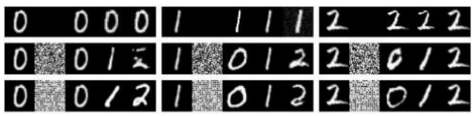
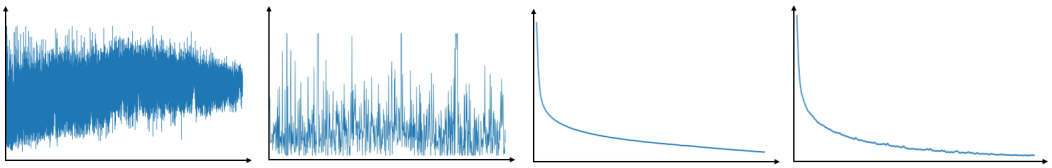


Figure 4: (Left) FID and AUB score for three images translation tasks in MNIST (lower the better for both metrics). FID score for each translation task is calculated by averaging scores from each direction and AUB score is shown in nats. This table shows that our model has overall better performances than all baselines models in terms of both metrics. AF in the first row indicates AlignFlow. (Right) Translation results among MNIST digits 0-2. Each block has first column as the original digit, the second as latent image, and the last three as translated results. Each row from the top indicates LRMF, AlignFlow, and ours respectively. Please note that second column of LRMF is set to be black because it does not have latent representation, and LRMF fails to translate in this situation which is why the numbers are all the same.

[1] under the terms of the Creative Commons Attribution-Share Alike 3.0 license



(a) discriminator loss in AlignFlow (b) validation metric in AlignFlow (c) training loss in AUB(ours) (d) validation loss in AUB(ours)

Figure 5: Training and validation loss comparisons. The objectives for AlignFlow trained with adversarial loss oscillates and is unstable during training while our loss plot (min-min) shows smooth convergence.

5 Discussion

Pros and cons of our method compared to adversarial method. Flow-based methods have different benefits and limitations compared to adversarial methods for distribution alignment. As one clear difference, our approach enables an application-agnostic yet theoretically grounded evaluation metric for comparing between models. Additionally, our min-min problem is fundamentally different than a min-max problem and avoids issues unique to min-max problems. However, our approach certainly has other limitations compared to GANs (e.g., it is restricted to invertible models and requires a density estimator). Overall, we ultimately want to convey our alignment approach is a feasible and fundamentally different alternative to adversarial, which is currently the only dominant approach. With our foundation, future work could focus on the performance aspects just as adversarial learning has been improving over the past seven years but is still an active area of research.

Benefit of AUB metric. Theoretically grounded dataset-agnostic metrics do not exist for GAN-based models [29, 30]. This is indeed an unsolved problem for distribution alignment via adversarial methods. Our AUB score provides a theoretically grounded metric that is dataset and domain agnostic and does not require any pretrained classifier. Specifically, it is an upper bound of GJSD between latent distributions up to a constant, which only depends on the training dataset—this is directly analogous to the fact that the negative log-likelihood is only equal to KL divergence up to a constant. Thus, while the optimal AUB is not zero, the *relative* performance of competing methods can be assessed, and *held-out test* AUB can be compared to train AUB to determine the model's generalization ability (even during training as seen in Fig. 5). We emphasize that our AUB score enables relative evaluation on tabular datasets, which is not usually considered because FID is inapplicable.

Extensibility. As mentioned in section 1 and section 2, our proposed idea is a general framework that can be harmonized with *any invertible flow models*, (e.g., Residual Flows [21], Flow++ [31]) and *any density models* (e.g., PixelCNN++ [24], FFJORD [20]). Hence, we argue our proposed idea is not limited to specific flow-based model or density model. We expect our framework would show better performance in domain alignment if it is combined with the aforementioned state-of-the-art components.

6 Conclusion

In this paper, we propose a novel variational upper bound on the generalized JSD that leads to a theoretically grounded alignment loss. We then show that this framework unifies previous flow-based approaches to distribution alignment and demonstrate the benefits of our approach compared to prior flow-based methods. In particular, our framework allows a straightforward extension to multi-distribution alignment that we show could be more parameter efficient than naively extending prior approaches. More broadly, we suggest that our AUB metric can be useful as a domain-agnostic metric for comparing distribution alignment methods (analogously to how ELBO is used to evaluate density estimation methods). An alignment metric that is not tied to a particular pretrained model (as for FID) or to a particular data type will be critical for systematic progress in unsupervised distribution alignment. We hope this paper provides one step in that direction.

References

[1] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

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

[3] 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)*, June 2018.

[4] 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.

[5] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In Sanjoy Dasgupta and David McAllester, editors, *International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 325–333, Atlanta, Georgia, USA, 17–19 Jun 2013.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

[7] 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.

[8] Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The GAN landscape: Losses, architectures, regularization, and normalization, 2019.

[9] Farzan Farnia and Asuman Ozdaglar. Do GANs always have Nash equilibria? In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3029–3039. PMLR, 13–18 Jul 2020.

[10] Weili Nie and Ankit B Patel. Towards a better understanding and regularization of gan training dynamics. In *Uncertainty in Artificial Intelligence*, pages 281–291. PMLR, 2020.

[11] Yue Wu, Pan Zhou, Andrew G Wilson, Eric Xing, and Zhiting Hu. Improving gan training with probability ratio clipping and sample reweighting. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5729–5740. Curran Associates, Inc., 2020.

[12] 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 I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[13] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On buggy resizing libraries and surprising subtleties in fid calculation. *arXiv preprint arXiv:2104.11222*, 2021.

[14] Ben Usman, Avneesh Sud, Nick Dufour, and Kate Saenko. Log-likelihood ratio minimizing flows: Towards robust and quantifiable neural distribution alignment. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21118–21129. Curran Associates, Inc., 2020.

[15] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, Nov 2021.

[16] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[17] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[18] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.

[19] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224, 2018.

[20] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[21] Ricky T. Q. Chen, Jens Behrmann, David K Duvenaud, and Joern-Henrik Jacobsen. Residual flows for invertible generative modeling. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[22] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2019.

[23] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

[24] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[25] Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

[26] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[27] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[28] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[29] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29:2234–2242, 2016.

[30] Ali Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

[31] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2722–2730. PMLR, 09–15 Jun 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We fixed random seed throughout our experiments.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]