# Spread Flows for Manifold Modelling

**Mingtian Zhang**
University College London*

**Yitong Sun**
Huawei Noah's Ark Lab

**Chen Zhang**
Huawei Noah's Ark Lab

**Steven McDonagh**
Huawei Noah's Ark Lab

## Abstract

Flow-based models typically define a latent space with dimensionality identical to the observational space. In many problems, however, the data does not populate the full ambient data space that they natively reside in, rather inhabiting a lower-dimensional manifold. In such scenarios, flow-based models are unable to represent data structures exactly as their densities will always have support off the data manifold, potentially resulting in degradation of model performance. To address this issue, we propose to learn a manifold prior for flow models that leverage the recently proposed *spread divergence* towards fixing the crucial problem; the KL divergence and maximum likelihood estimation are ill-defined for manifold learning. In addition to improving both sample quality and representation quality, an auxiliary benefit enabled by our approach is the ability to identify the intrinsic dimension of the manifold distribution.

## 1 Introduction

Normalizing flows Rezende and Mohamed (2015) have shown considerable potential for the task of modelling and inferring expressive distributions through the learning of well-specified probabilistic models. Recent progress in this area has defined a set of general and extensible structures, capable of representing highly complex and multimodal distributions, see Kobyzev et al. (2020) for a detailed overview. Specifically, assume an absolutely continuous [1] (a.c.) random variable (r.v.) $Z$ with distribution $\mathbb{P}_Z$ and probability density $p_Z(z)$. We can transform $Z$ to get a r.v. $X$: $X = f(Z)$, where $f : \mathbb{R}^D \to \mathbb{R}^D$ is an invertible function with inverse $f^{-1} = g$, so $X$ has a (log) density function $p_X(x)$ with the following form

$$\log p_X(x) = \log p_Z(g(x)) + \log \left| \det \left( \frac{\partial g}{\partial x} \right) \right|, \quad (1)$$

where $\log \left| \det \left( \frac{\partial g}{\partial x} \right) \right|$ is the log determinant of the Jacobian matrix. We call $f$ (or $g$) a *volume-preserving* function if the log determinant is equal to 0. Training of flow models typically makes use of MLE. We assume the data random variable $X_d$ with distribution $\mathbb{P}_d$ is a.c. and has density $p_d(x)$. In addition to the well-known connection between MLE and minimization of the KL divergence $\mathrm{KL}(p_d(x)||p_X(x))$ in $X$ space (see Appendix A for details), MLE is also equivalent to minimizing the KL divergence in $Z$ space, due to the KL divergence invariance under invertible transformations Yeung (2008); Papamakarios et al. (2019). Specifically, we define $Z_{\mathbb{Q}} : Z_{\mathbb{Q}} = g(X_d)$ with distribution $\mathbb{Q}_Z$ and density function[2] $q(z)$, the KL divergence in $Z$ space $\mathrm{KL}(q(z)||p(z))$ can be written as

$$-\int p_d(x) \left( \log p_Z\left(g(x)\right) + \log \left| \det \left( \frac{\partial g}{\partial x} \right) \right| \right) dx, \quad (2)$$

with constant term discarded, the full derivation can be found in Appendix A. Since we can only access samples $x_1, x_2, \ldots, x_N$ from $p_d(x)$, we approximate the integral by Monte Carlo sampling

$$\mathrm{KL}(q(z)||p(z)) \approx -\frac{1}{N} \sum_{n=1}^{N} \log p_X(x_n) + const.. \quad (3)$$

We highlight the connection between MLE and KL divergence minimization in $Z$ space for flow models. The prior distribution $p(z)$ is usually chosen to be a $D$-dimensional Gaussian distribution.

*This work was done during an internship in Huawei Noah's Ark Lab. Correspondence to: Mingtian Zhang m.zhang@cs.ucl.ac.uk.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. 

[1] The distribution is a.c. with respect to the Lebesgue measure, so it has a density function, see Durrett (2019).

[2] Since we assume $X_d$ is a.c., $Z_{\mathbb{Q}} : Z_{\mathbb{Q}} = g(X_d)$ is also a.c. with a bijiective mapping $g(\cdot)$ and thus $Z_{\mathbb{Q}}$ allows a density function.

Further to this, we note that if data lies on a lower-dimensional manifold, and thus does not populate the full ambient space, then the estimated flow model will necessarily have mass lying off the data manifold, which may result in under-fitting and poor generation qualities. Contemporary flow-based approaches may make for an inappropriate representation choice in such cases. Formally, when data distribution $\mathbb{P}_d$ is singular, *e.g.* a measure on a low dimensional manifold, then $\mathbb{P}_d$ or the induced latent distribution $\mathbb{Q}_Z$ will no longer emit valid density functions. In this case, the KL divergence and MLE in equation 2 are typically not well-defined under the considered flow model assumptions. This issue brings theoretical and practical challenges that we discuss in the next section.

## 2 Flow Models for Manifold Data

We assume a data sample $\mathbf{x} \sim \mathbb{P}_d$ to be a $D$ dimensional vector $\mathbf{x} \in \mathbb{R}^D$ and define the ambient dimensionality of $\mathbb{P}_d$, denoted by $\texttt{Amdim}(\mathbb{P}_d)$, to be $D$. However for many datasets of interest, *e.g.* natural images, the data distribution $\mathbb{P}_d$ is commonly believed to be supported on a lower dimensional manifold Beymer and Poggio (1996). We assume the dimensionality of the manifold to be $K$ where $K < D$, and define the intrinsic dimension of $\mathbb{P}_d$, denoted by $\texttt{Indim}(\mathbb{P}_d)$, to be the dimension of this manifold. Figure 1a provides an example of this setting where $\mathbb{P}_d$ is a 1D distribution in 2D space. Specifically, each data sample $\mathbf{x} \sim \mathbb{P}_d$ is a 2D vector $\mathbf{x} = \{x_1, x_2\}$ where $x_1 \sim \mathcal{N}(0, 1)$ and $x_2 = \sin(2x_1)$. Therefore, this example results in $\texttt{Amdim}(\mathbb{P}_d) = 2$ and $\texttt{Indim}(\mathbb{P}_d) = 1$.

In flow-based models, function $f$ is constructed such that it is both bijective and differentiable. When the prior $\mathbb{P}_Z$ is a distribution whose support is $\mathbb{R}^D$ (*e.g.* Multivariate Gaussian distribution), the marginal distribution $\mathbb{P}_X$ will also have support $\mathbb{R}^D$ and $\texttt{Amdim}(\mathbb{P}_X) = \texttt{Indim}(\mathbb{P}_X) = D$. When the support of the data distribution lies on a $K$-dimensional manifold and $K < D$, $\mathbb{P}_d$ and $\mathbb{P}_X$ are constrained to have different support. That is, the intrinsic dimensions of $\mathbb{P}_X$ and $\mathbb{P}_d$ are always different; $\texttt{Indim}(\mathbb{P}_X) \neq \texttt{Indim}(\mathbb{P}_d)$. In this case it is impossible to learn a model distribution $\mathbb{P}_X$ identical to the data distribution $\mathbb{P}_d$. Nevertheless, flow-based models have shown strong empirical success in real-world problem domains such as the ability to generate high-quality and realistic images Kingma and Dhariwal (2018). Towards investigating the cause, and explaining this disparity between theory and practice, we employ a toy example to provide intuition for the effects and consequences resulting from model and data distributions that possess differing intrinsic dimensions.

Consider the toy dataset introduced previously; a 1D distribution lying in a 2D space (Figure 1a). The prior density $p(z)$ is a standard 2D Gaussian $p(z) = \mathcal{N}(0, I_Z)$ and the function $f$ is a non-volume preserving flow with two coupling layers (see Appendix C.1). In Figure 1b we plot samples from the flow model; the sample $\mathbf{x}$ is generated by first sampling a 2D datapoint $\mathbf{z} \sim \mathcal{N}(0, I_Z)$ and then letting $\mathbf{x} = f(\mathbf{z})$. Figure 1c shows samples from the prior distributions $\mathbb{P}_Z$ and $\mathbb{Q}_Z$. $\mathbb{Q}_Z$ is defined as the transformation of $\mathbb{P}_d$ using the bijective function $g$, such that $\mathbb{Q}_Z$ is constrained to support a 1D manifold in 2D space, and $\texttt{Indim}(\mathbb{Q}_Z) = \texttt{Indim}(\mathbb{P}_d) = 1$. Training of $\mathbb{Q}_Z$ to match $\mathbb{P}_Z$ (which has intrinsic dimension 2), can be seen to result in "curling up" of the manifold in the latent space, contorting it towards satisfying a distribution that has intrinsic dimension 2 (see Figure 1c). This ill-behaved phenomenon causes several potential problems for contemporary flow models:

1. *Poor sample quality.* Figure 1b shows examples where incorrect assumptions result in the model generating poor samples.

2. *Low quality data representations.* The discussed characteristic that results in "curling up" of the latent space may cause representation quality degradation.

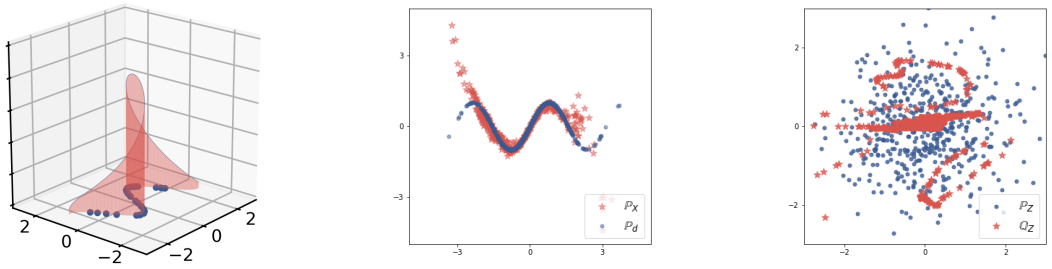
3. *Inefficient use of network capacity*. Neural network capacity is spent on contorting the distribution $\mathbb{Q}_Z$ to satisfy imposed dimensionality constraints.

A natural solution to the problem of intrinsic dimension mismatch is to select a prior distribution $\mathbb{P}_Z$ with the same dimensionality as the intrinsic dimension of the data distribution such that: $\texttt{Indim}(\mathbb{P}_Z) = \texttt{Indim}(\mathbb{P}_d)$. However, since the $\texttt{Indim}(\mathbb{P}_d)$ is unknown, one option is to instead learn it from the data. In the next section, we introduce an approach that enables us to learn $\texttt{Indim}(\mathbb{P}_d)$.

## 3 Learning a Manifold Prior

Consider a data vector $\mathbf{x} \in \mathbb{R}^D$, then a flow-based model prior $\mathbb{P}_Z$ is usually given by a $D$-dimensional Gaussian distribution or alternative simple distribution that is also absolutely continuous (a.c.) in $\mathbb{R}^D$. Therefore, the intrinsic dimension $\texttt{Indim}(\mathbb{P}_Z) = D$. To allow a prior to have an intrinsic dimension strictly less than $D$, we let $\mathbb{P}_Z$ be a '*generalized Gaussian*'[3] distribution $\mathcal{GN}(0, AA^T)$, where $\mathbf{z} \in \mathbb{R}^D$ and $A$ is a $D \times D$ lower triangular matrix with $D(D+1)/2$ parameters, such that $AA^T$ is constrained to be a positive semi-definite matrix. When $AA^T$ has full rank $D$, then $\mathbb{P}_Z$ is a (non-degenerate) multivariate Gaussian on $\mathbb{R}^D$. When $\texttt{Rank}(AA^T) = K$ and $K < D$, then $\mathbb{P}_Z$ will degenerate to a Gaussian supported on a $K$-dimensional manifold, such that the intrinsic dimension

[3] We use the generalized Gaussian to include the case that the covariance $AA^T$ is not full rank.



(a) $\mathbb{P}_d$ (b) $X$ space (c) $Z$ space

Figure 1: Samples and latent visualization from a flow based model with a fixed Gaussian prior where the intrinsic dimension is strictly lower than the true dimensionality of the data space.

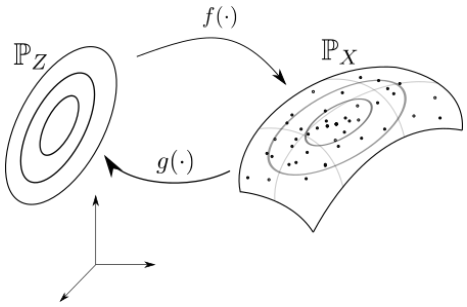



Figure 2: This figure gives an overview of our method. For data (black dots) that lie on a 2D manifold in 3D space, we want to use a model $\mathbb{P}_X$ with $\texttt{Indim}(\mathbb{P}_X) = 2$. Therefore, we learn a prior $\mathbb{P}_Z$ that is a 2D Gaussian in 3D space and an invertible function $f$ which maps from $\mathbb{P}_Z$ to $\mathbb{P}_X$.

$\texttt{Indim}(\mathbb{P}_Z) = K$[4]. Figure 2 illustrates a sketch of this scenario. In practice, we initialize $A$ to be an identity matrix, thus $AA^T$ is also an identity and $\mathbb{P}_Z$ is initialized as a standard Gaussian. We want to highlight that the classic flow method with a standard Gaussian prior is a special case with $A = I$ in our model and the additional free parameters in matrix $A$ allow the model to capture the manifold property of the target data distribution.

**Identification of the Intrinsic Dimension** A byproduct of our model is that the intrinsic dimension of the data manifold can be identified. When the model matches the true data distribution $\mathbb{P}_X = \mathbb{P}_d$, the supports of $\mathbb{P}_X$ and $\mathbb{P}_d$ will also have the same intrinsic dimension: $\texttt{Indim}(\mathbb{P}_X) = \texttt{Indim}(\mathbb{P}_d)$. The flow function $f$, and its inverse $g = f^{-1}$, are bijective and continuous so $f$ is a diffeomorphism Kobyzev et al. (2020). Due to the *invariance of dimension* property of diffeomorphisms Lee (2013), the manifold that supports $\mathbb{P}_Z$ will have the same dimension as the manifold that supports $\mathbb{P}_X$ thus

$$\texttt{Indim}(\mathbb{P}_Z) = \texttt{Indim}(\mathbb{P}_X) = \texttt{Indim}(\mathbb{P}_d). \quad (4)$$

Since the intrinsic dimension of $\mathbb{P}_Z$ is equal to the rank of the matrix $AA^T$, we can calculate $\texttt{Rank}(AA^T)$ by counting the number of non-zero eigenvalues of the matrix $AA^T$. This allows for identification of the intrinsic dimension of the data distribution as

$$\texttt{Indim}(\mathbb{P}_d) = \texttt{Rank}(AA^T). \quad (5)$$

**Dimension Reduction** We would also like to conduct dimension reduction using the learned manifold flow model. For $\mathbb{Q}_Z$ with $\texttt{Amdim}(\mathbb{Q}_Z)$=$D$ and $\texttt{Indim}(\mathbb{Q}_Z)$=$K$, we first conduct an eigen-value decomposition of the $D \times D$ matrix $AA^T$: $AA^T = \mathbf{E}\Lambda\mathbf{E}^T$, where $\mathbf{E} = [e_1, \cdots, e_D]$ contains all the eigen-vectors and $\Lambda = \text{diag}(\lambda_1, \cdots, \lambda_D)$. When $\texttt{Rank}(AA^T) = K \leq D$, there exist $K$ eigenvectors with positive eigenvalues. We select the first $K$ eigenvectors and form the matrix $\mathbf{E} = [\mathbf{e}^1, \ldots, \mathbf{e}^K]$ with dimension $D \times K$. We then transform each data sample $\mathbf{x} \in \mathbb{R}^D$ into $Z$ space: $\mathbf{z} = g(\mathbf{x})$, such that $\mathbf{z} \in \mathbb{R}^D$. Finally, a linear projection is carried out $\mathbf{z}^{proj} = \mathbf{z}\mathbf{E}$ to obtain the lower dimensional representation $\mathbf{z}^{proj} \in \mathbb{R}^K$. This procedure can be seen as a *nonlinear PCA*, where the nonlinearity is learned by the inverse flow function $g$.

**Sample from the Manifold Flow** To sample from the flow model with $\texttt{Indim}(\mathbb{P}_X)$, we can first take a sample $\epsilon$ from a standard $K$-dimensional Gaussian distribution and let $\mathrm{z}' = \mathbf{E}_K\sqrt{\Lambda_K}$ where $\mathbf{E}_K$ are the first $K$ eigenvectors and $\Lambda_K$ is the corresponding eigen-values. When $K = D$, the linear matrix $\mathbf{E}_K\sqrt{\Lambda_K}$ will be equivalent to the Cholesky decomposition solution of the matrix $AA^T$. We can then let $\mathrm{x}' = f(\mathrm{z}')$ as the sample from the model.

## 4 Addressing Ill-defined KL divergence

When $\texttt{Rank}(AA^T) < D$, the degenerate covariance $AA^T$ is no longer invertible and we are unable to evaluate the

[4]By the definition of the manifold Lee (2013), the intrinsic dimension of a manifold is equal or smaller than its ambient dimension.

density value of $p(\mathbf{z})$ for a given random vector $\mathbf{z}$. Furthermore, when the data distribution $\mathbb{P}_d$ is supported on a $K$-dimensional manifold, $\mathbb{Q}_Z$ will also be supported on a $K$-dimensional manifold and no longer has a valid density function. Using equation 2 to train the flow model then becomes impossible as the KL divergence between $\mathbb{P}_Z$ and $\mathbb{Q}_Z$ is not well defined[5]. Recent work by Zhang et al. (2020, 2019) proposed a new family of divergence to address this problem, which we introduce below.

Let $Z_{\mathbb{Q}}$ and $Z_{\mathbb{P}}$ be two random variables with distributions $\mathbb{Q}_Z$ and $\mathbb{P}_Z$, respectively. The KL divergence between $\mathbb{Q}_Z$ and $\mathbb{P}_Z$ is not well defined if $\mathbb{Q}_Z$ or $\mathbb{P}_Z$ does not have valid density functions. Let $K$ be an a.c. random variable that is independent of $Z_{\mathbb{Q}}$ and $Z_{\mathbb{P}}$ and has density $p_K$, We define $Z_{\tilde{\mathbb{P}}} = Z_{\mathbb{P}} + K; Z_{\tilde{\mathbb{Q}}} = Z_{\mathbb{Q}} + K$ with distributions $\tilde{\mathbb{P}}_Z$ and $\tilde{\mathbb{Q}}_Z$ respectively. Then $\tilde{\mathbb{P}}_Z$ and $\tilde{\mathbb{Q}}_Z$ are a.c. (Durrett, 2019, Theorem 2.1.16) with density functions

$$q(\tilde{\mathbf{z}}) = \int_{\mathbf{z}} p_K(\tilde{\mathbf{z}} - \mathbf{z}) d\mathbb{Q}_Z, \quad p(\tilde{\mathbf{z}}) = \int_{\mathbf{z}} p_K(\tilde{\mathbf{z}} - \mathbf{z}) d\mathbb{P}_Z. \quad (6)$$

The *spread KL divergence* between $\mathbb{Q}_Z$ and $\mathbb{P}_Z$ as the KL divergence between $\tilde{\mathbb{Q}}_Z$ and $\tilde{\mathbb{P}}_Z$ is:

$$\widetilde{\mathrm{KL}}(\mathbb{Q}_Z || \mathbb{P}_Z) \equiv \mathrm{KL}(\tilde{\mathbb{Q}}_Z || \tilde{\mathbb{P}}_Z) \equiv \mathrm{KL}\left(q(\tilde{\mathbf{z}}) || p(\tilde{\mathbf{z}})\right). \quad (7)$$

In this work we let $K$ be a Gaussian with diagonal covariance $\sigma_Z^2 I$ to satisfy the sufficient conditions such that $\widetilde{\mathrm{KL}}(\cdot)$ is a valid divergence (see Zhang et al. (2020) for details) and has the properties:

$$\widetilde{\mathrm{KL}}(\mathbb{Q}_Z || \mathbb{P}_Z) \geq 0, \quad \widetilde{\mathrm{KL}}(\mathbb{Q}_Z || \mathbb{P}_Z) = 0 \Leftrightarrow \mathbb{Q}_Z = \mathbb{P}_Z. \quad (8)$$

Since $\mathbb{Q}_Z$ and $\mathbb{P}_Z$ are transformed from $\mathbb{P}_d$ and $\mathbb{P}_X$ using an invertible function $g$, we have

$$\mathbb{Q}_Z = \mathbb{P}_Z \Leftrightarrow \mathbb{P}_d = \mathbb{P}_X. \quad (9)$$

Therefore, the spread KL divergence can be used to train flow-based models with a manifold prior in order to fit a dataset that lies on a lower-dimensional manifold.

### 4.1 Estimation of the Spread KL Divergence

As shown in equation 7, minimizing $\widetilde{\mathrm{KL}}(\mathbb{Q}_Z || \mathbb{P}_Z)$ is equivalent to minimizing $\mathrm{KL}(q(\tilde{\mathbf{z}}) || p(\tilde{\mathbf{z}}))$, which has two terms

$$\widetilde{\mathrm{KL}}(\mathbb{Q}_Z || \mathbb{P}_Z) = \underbrace{\int q(\tilde{\mathbf{z}}) \log q(\tilde{\mathbf{z}}) d\tilde{\mathbf{z}}}_{\textbf{Term 1}} - \underbrace{\int q(\tilde{\mathbf{z}}) \log p(\tilde{\mathbf{z}}) d\tilde{\mathbf{z}}}_{\textbf{Term 2}}, \quad (10)$$

where $q(\tilde{\mathbf{z}})$ and $p(\tilde{\mathbf{z}})$ are defined in equation 6. We next discuss the estimation of this objective.

**Term 1:** We use $\mathrm{H}(\cdot)$ to denote the differential entropy. Term 1 is the denoted as $-\mathrm{H}(Z_{\tilde{\mathbb{Q}}})$. For a volume-preserving $g$ and an a.c. $X_d$, the entropy $\mathrm{H}(Z_{\mathbb{Q}}) = \mathrm{H}(X_d)$ is independent of the model parameters and can be ignored during training. However, the entropy $\mathrm{H}(Z_{\tilde{\mathbb{Q}}}) = \mathrm{H}(Z_{\mathbb{Q}} + K)$ will still depend on $g$, see Appendix B.1 for an example. We claim that when the variance of $K$ is small, the dependency between $\mathrm{H}(Z_{\tilde{\mathbb{Q}}})$ and the volume preserving function $g$ is weak, thus we can approximate equation 10 by disregarding term 1 and this will not adversely affect the training.

To build intuitions, we assume $X_d$ is a.c., so $Z_{\mathbb{Q}} = g(X_d)$ is also a.c.. Using standard entropic properties Kontoyiannis and Madiman (2014), we have the following relationship

$$\mathrm{H}(Z_{\mathbb{Q}}) \leq \mathrm{H}(Z_{\mathbb{Q}} + K) = \mathrm{H}(Z_{\mathbb{Q}}) + \mathrm{I}(Z_{\mathbb{Q}} + K, K), \quad (11)$$

where $\mathrm{I}(\cdot, \cdot)$ denotes the mutual information. Since $Z_{\mathbb{Q}}$ is independent of function $g$ and if $\sigma_Z^2 \to 0$, then $\mathrm{I}(Z_{\mathbb{Q}} + K, K) \to 0$ (see Appendix B.2 for a proof), the contribution of the $\mathrm{I}(Z_{\mathbb{Q}} + K, K)$ term, with respect to training $g$, becomes negligible in the case of small $\sigma_Z^2$.

Unfortunately, equation 11 is no longer valid when $\mathbb{P}_d$ lies on a manifold since $Z_{\mathbb{Q}}$ will be a singular random variable and the differential entropy $\mathrm{H}(Z_{\mathbb{Q}})$ is not defined. In Appendix B.3, we show that leaving out the entropy $\mathrm{H}(Z_{\tilde{\mathbb{Q}}})$ corresponds to minimizing an *upper bound* of the spread KL divergence. To further explore the contribution of the negative entropy term, we compare leaving out $-\mathrm{H}(Z_{\tilde{\mathbb{Q}}})$ with alternatively approximating $-\mathrm{H}(Z_{\tilde{\mathbb{Q}}})$ during training. In Appendix B.4, we discuss an appropriate approximation technique and provide empirical evidence which shows that ignoring $-\mathrm{H}(Z_{\tilde{\mathbb{Q}}})$ will not adversely affect the training of our model. Therefore, we make use of volume preserving $g$ and small variance $\sigma_Z^2 = 1 \times 10^{-4}$ in our experiments.

In contrast to other volume-preserving flows, that utilize a fixed prior $\mathbb{P}_Z$, our method affords additional flexibility by way of allowing for changes to the 'volume' of the prior towards matching the distribution of the target data. In this way, our decision to employ volume-preserving flow functions does not limit the expressive power of the model, in principle. Popular non-volume preserving flow structures, *e.g.* affine coupling flow, may also easily be normalized to become volume preserving, thus further extending the applicability of our approach (see Appendix C.1).

**Term 2:** The noisy prior $p(\tilde{\mathbf{z}})$ is defined to be a degenerate Gaussian $\mathcal{N}(0, AA^T)$, convolved with Gaussian noise $\mathcal{N}(0, \sigma_Z^2 I)$, and has a closed form density

$$p(\tilde{\mathbf{z}}) = \mathcal{N}(0, AA^T + \sigma_Z^2 I). \quad (12)$$

Therefore, the log density $\log p(\tilde{\mathbf{z}})$ is well defined, we can approximate term 2 using Monte Carlo

$$\int q(\tilde{\mathbf{z}}) \log p(\tilde{\mathbf{z}}) d\tilde{\mathbf{z}} \approx \frac{1}{N} \sum_{n=1}^{N} \log p(\tilde{\mathbf{z}}_n), \quad (13)$$

where $q(\tilde{\mathbf{z}}) = \int p(\tilde{\mathbf{z}}|\mathbf{z}) d\mathbb{Q}_Z$. To sample from $q(\tilde{\mathbf{z}})$, we first get a data sample $\mathbf{x} \sim \mathbb{P}_d$, use function $g$ to get $\mathbf{z} = g(\mathbf{x})$

[5]The KL divergence $\mathrm{KL}(\mathbb{Q}||\mathbb{P})$ is well defined when $\mathbb{Q}$ and $\mathbb{P}$ have valid densities with common support Ali and Silvey (1966).

(so $\mathbf{z}$ is a sample of $\mathbb{Q}_Z$) and finally sample $\tilde{\mathbf{z}} \sim p(\tilde{\mathbf{z}}|\mathbf{z})$. The training details can be found in Algorithm 1.

**Algorithm 1** Training Spread Flows

**Given:** $\mathcal{X}_{train} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\} \sim \mathbb{P}_d, \mathcal{N}(0, \sigma_Z^2 I)$
**Model:** An inverse flow function $g$, a lower triangular matrix $A$ initialized as an identity matrix.
**while** not converge **do**
Sample a data batch $\mathcal{X}_B = \{\mathbf{x}_1, \cdots, \mathbf{x}_B\} \in \mathcal{X}_{train}$.
Transform data $z_b = g(\mathbf{x}_b)$ for $\mathbf{x}_b \in \mathcal{X}_B$.
Sample noise $\epsilon_b \sim \mathcal{N}(0, \sigma_Z^2 I)$ for each $\mathbf{x}_b$.
Add noise $\tilde{\mathbf{z}}_b = \mathbf{z}_b + \epsilon_b$ for each latent representation.
Train $g_\theta$ and $A$ using Equation 13.
**end while**

## 5 Experiments

Traditional flows assume the data distributions are a.c. and we extend this assumption such that the data distribution lies on one continuous manifold. We demonstrate both the effectiveness and robustness of our model by considering firstly (1) data distributions that satisfy our continuous manifold assumption: toy 2D and 3D data, the *fading square* dataset; and secondly (2) distributions where our assumption no longer holds: synthesized and real MNIST LeCun (1998), and CelebA Liu et al. (2015a). We use incompressible affine coupling layers, introduced by Sorrenson et al. (2020); Dinh et al. (2016), for toy and MNIST experiments and a volume-preserving variation of the Glow structure Kingma and Dhariwal (2018). See Appendix C for further experimental details.

### 5.1 Synthetic Data

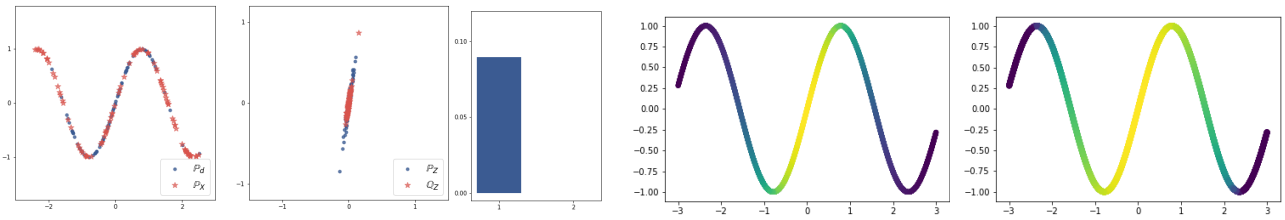
**2D toy data** We firstly verify our method using the toy data depicted in Figure 1a. Figure 3 shows model samples, the learned prior and the eigenvalues of $AA^T$. We observe that sample quality improves upon those in Figure 1b and the prior has learned a degenerate Gaussian with $\texttt{Indim}(\mathbb{P}_Z) = 1$, matching $\texttt{Indim}(\mathbb{P}_d)$. We also highlight that our model, in addition to learning the manifold support of the target distribution, can capture the 'density' allocation on the manifold, see Appendix C.2 for further details.

**3D toy data (S-curve)** We model the S-curve dataset (Figure 4a), where the data lies on a 2D manifold in a 3D space ($\texttt{Indim}(\mathbb{P}_d)$=2), see Appendix C.1 for details. Our model learns a function $g$ to transform $\mathbb{P}_d$ to $\mathbb{Q}_Z$, where the latter lies on a 2D linear subspace in 3D space (see Figure 4c). As discussed in Section 3, we also conduct a linear dimension reduction to generate 2D representations, see Figure 4f. The colormap indicates correspondence between the data in 3D space and the 2D representation. Our method can be observed to successfully: (1) identify the intrinsic dimensionality of the data and (2) project the data into a 2D space that faithfully preserves the structure of the original distribution. In contrast, we find that a flow with fixed Gaussian prior fails to learn such a data distribution and generate meaningful representations, see Figure 4g and 4h.

**Fading Square dataset** The *fading square* dataset Rubenstein et al. (2018) was proposed in order to assess model behavior when data distribution and model possess differing intrinsic dimension and therefore affords a relevant test of our work. The dataset consists of 32×32 images with 6×6 grey squares on a black background. The grey scale values are sampled from a uniform distribution with range $[0, 1]$, so $\texttt{Indim}(\mathbb{P}_d)$=1. Figure 5a shows samples from the dataset to which we fit our model and additional model details may be found in Appendix C. Figure 5b shows samples from our trained model and Figure 5d shows the first 20 eigenvalues of $AA^T$ (ranked from high to low), we observe that only one eigenvalue is larger than zero and the others have converged to zero. This illustrates we have successfully identified the intrinsic dimension of $\mathbb{P}_d$. We further carry out the dimensionality reduction process; the latent representation $\mathbf{z}$ is projected onto a 1D line and we plot the correspondence between the projected representation and the data in Figure 5e. Pixel values can be observed to decay as the 1D representation is traversed from left to right, indicating the representations are consistent with the properties of the original data distribution. In contrast, we find that the traditional flow model, with fixed 1024D Gaussian prior, fails to learn the data distribution; see Figure 5c.

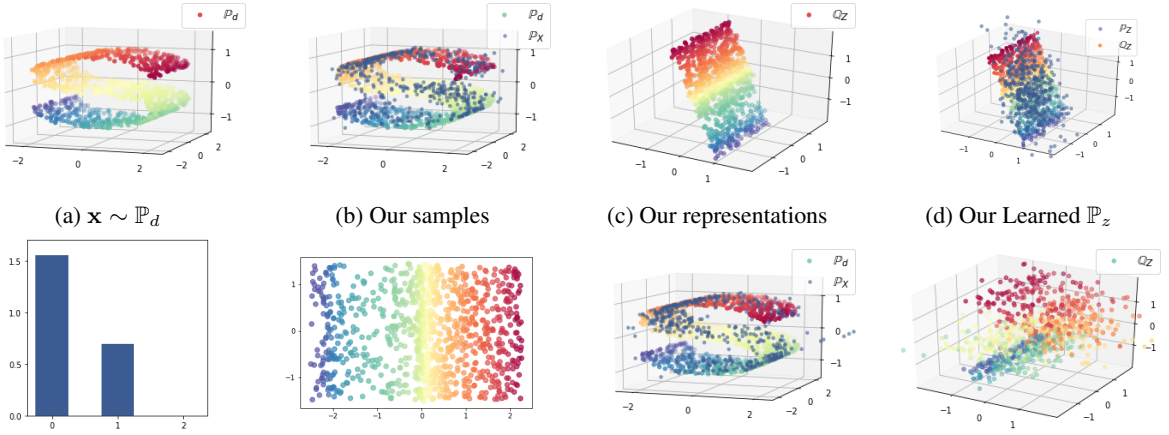
**Synthetic MNIST** We further investigate model training using digit images. It may be noted that, for real-world image datasets, the true intrinsic dimension is unknown. Therefore in order to further verify the correctness of our model's ability to identify intrinsic dimension, we first construct a synthetic dataset by fitting an implicit model $p_\theta(\mathbf{x}) = \int \delta(\mathbf{x} - g(\mathbf{z}))p(\mathbf{z}))d\mathbf{z}$ to the MNIST, and then sample from the trained model $\mathbf{x} \sim p_\theta(\mathbf{x})$ in order to generate synthetic training data. The intrinsic dimension of the training data can then be approximated[6] via the dimension of the latent variable $Z$: $\texttt{Indim}(\mathbb{P}_d)$=$\texttt{dim}(Z)$. We construct two datasets with $\texttt{dim}(Z)$=5 and $\texttt{dim}(Z)$=10 such that $\texttt{Indim}(\mathbb{P}_d)$=5 and $\texttt{Indim}(\mathbb{P}_d)$=10, respectively. Since the generator of the implicit model is not constrained to be invertible, a diffeomorphism between data distribution and prior will not hold and will also not necessarily lie on a continuous manifold. We found when the continuous manifold assumption is not satisfied, training instabilities can occur. Towards alleviating this issue, we smooth the

[6]For implicit model: $X = g(Z)$, the intrinsic dimension of the model distribution will be not-greater-than the dimension of the latent variable: $\texttt{Indim}(X) \leq \texttt{dim}(Z)$ Arjovsky and Bottou (2017b). Further, if strictly $\texttt{Indim}(X) < \texttt{dim}(Z)$ this results in a 'degenerated' case. When we fit the model on a data distribution where $\texttt{Indim}(X_d) \geq \texttt{dim}(Z)$, we assume that degeneration will not occur during training, resulting in $\texttt{Indim}(\mathbb{P}_d) \approx \texttt{dim}(Z)$. For simplicity, we equate $\texttt{Indim}(\mathbb{P}_d)$=$\texttt{dim}(Z)$ in the main text.



(a) $X$ space (b) $Z$ space (c) Eigenval. (d) True Density (e) Our Estimation

Figure 3: (a) shows the samples from $\mathbb{P}_d$ (blue) and model $\mathbb{P}_X$ (red). (b) shows the samples from the learned prior $\mathbb{P}_Z$ (blue) and $\mathbb{Q}_Z$ (red). (c) shows the eigenvalues of $AA^T$. (d) and (e) show the true and the learned density on the manifold.



(a) $\mathbf{x} \sim \mathbb{P}_d$ (b) Our samples (c) Our representations (d) Our Learned $\mathbb{P}_z$

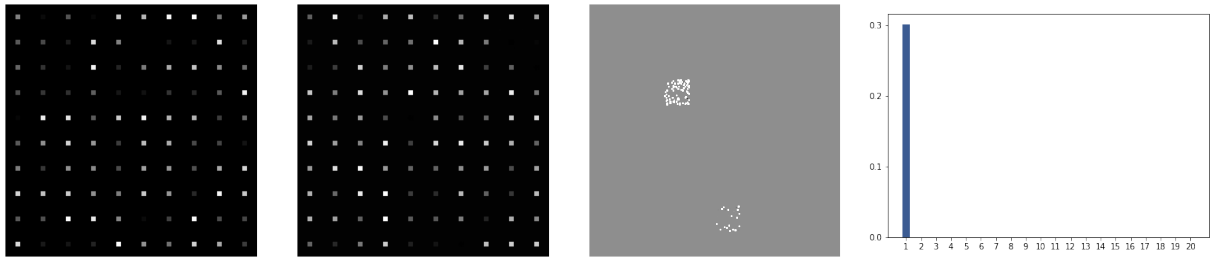
(e) Eigenvalues (f) $\mathbf{z}^{proj} = \mathbf{zE}$ (g) Trad. Flow samples (h) Trad. Flow representations

Figure 4: (a) S-curve data $\mathbf{x} \sim \mathbb{P}_d$. (b) Samples generated from our learned model. (c) Latent representation $\mathbf{z} = g(\mathbf{x})$, points are lying on a linear subspace. (d) Samples from the learned prior distribution. (e) Eigenvalues of the matrix $AA^T$, we deduce that $\texttt{Indim}(\mathbb{P}_d) = 2$. (f) Our representation after the dimensionality reduction $\mathbf{z}^{proj} = \mathbf{zE}$. (g) and (h): Samples and representations from a learned traditional flow model with a fixed Gaussian prior.

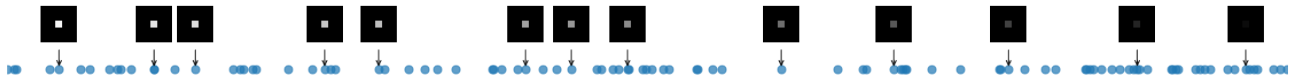
data manifold by adding small Gaussian noise (with standard deviation $\sigma_x$=0.05) to the training data. We note that adding Gaussian noise changes the intrinsic dimension of the training distribution towards alignment with the ambient dimension and therefore we mitigate this undesirable effect by annealing $\sigma_x$ after $2000k$ iterations with a factor of $0.9$ every $10k$ iteration. Experimentally, we cap a lower-bound Gaussian noise level of $0.01$ to help retain successful model training and find the effect of the small noise to be negligible when estimating the intrinsic dimension. In Figure 6, we plot the first twenty eigenvalues of $AA^T$ (ranked from high to low) after fitting two synthetic MNIST with $\texttt{dim}(\mathbb{P}_d)$=$\{5, 10\}$. We can find that 5 and 10 eigenvalues are significantly larger than other values, respectively. The remaining non-zero eigenvalues can be attributed to the added small Gaussian noise. Therefore, our method can successfully model complex data distributions and identify the intrinsic dimension when the data distribution lies on a continuous manifold. We also provide model sample comparisons in Figure 7, where we can find our model can achieve better sample quality compared to the traditional flow. In the next section, we apply our model to real image data where this assumption may not be satisfied.

### 5.2 Real World Data

**Real MNIST** For the real MNIST, it was shown that digits have differing intrinsic dimensions Costa and Hero (2006). This suggests the MNIST may lie on *several*, *disconnected* manifolds with differing intrinsic dimensions. Although our model assumes that $\mathbb{P}_d$ lies on one continuous manifold, it is interesting to investigate the case when this assumption is not fulfilled. We thus fit our model to the original MNIST and plot the eigenvalues in Figure 6 (a) and (b). In contrast to Figures 6a, 6b; the gap between eigenvalues predictably exhibits a less obvious step change. However, the values suggest the intrinsic dimension of MNIST lies between 11 and 14. This result is consistent with previous estimations, stating that the intrinsic dimension of MNIST is between 12 and 14 Facco et al. (2017); Hein and Audib-



(a) $\mathbf{x} \sim \mathbb{P}_d$ (b) Our model (c) Traditional flow (d) Eigenvalues of $AA^T$



(e) $\mathbf{z}^{proj} = \mathbf{z}\mathbf{E}$

Figure 5: (a) and (b) show samples from $\mathbb{P}_d$ and our model, respectively. (c) shows a traditional flow based model with a fixed Gaussian prior fails to fit the data distribution and cannot generate valid samples. (d) the first 20 eigenvalues of the matrix $AA^T$. (e) shows the representation after applying dimensionality reduction. See text for further discussion.

ert (2005); Costa and Hero (2006), see also Figure 7 for the model samples comparisons. Recent work Cornish et al. (2019) discusses fitting flow models to a $\mathbb{P}_d$ that lies on disconnected components, by introducing a mixing prior. Such techniques may be easily combined with our method towards constructing more powerful flow models.

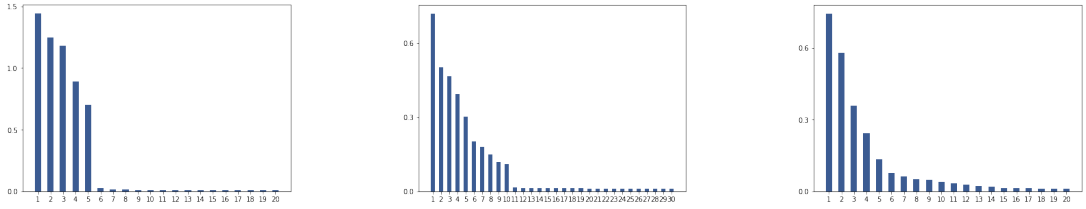
**CelebA** To model CelebA, we first centre-crop and resize each image to $32\times32\times3$. Pixels take discrete values from $[0, \dots, 255]$ and we follow a standard dequantization process Uria et al. (2013): $x{=}(x + \text{Uniform}(0, 1))/256$ to transform each pixel to a continuous value in $[0, 1]$. Similar to the MNIST experiment, we add small Gaussian noise (with standard deviation $\sigma_x{=}0.01$) for twenty initial training epochs in order to smooth the data manifold and then anneal the noise with a factor of $0.9$ for a further twenty epochs. Additional training and network structure details can be found in Appendix C.6.

We aim to empirically verify if the learned model is concentrated on a low-dimensional manifold and examine the validity of the estimated intrinsic dimensions. Following Ross and Cresswell (2021), we study if the data can be reconstructed from a low-dimensional linear manifold specified by $AA^T$. In Figure 9, we plot the learned eigenvalues $\Lambda$ in log space and observe that eigenvalues have an exponential decay starting at $\sim$2700, which suggests that the intrinsic dimension of the CelebA is around 2700. For reconstructions, we first calculate the low dimensional representation of image $\mathbf{x}$ by transforming it to the latent space with the inverse flow $\mathbf{z} = g(\mathbf{x})$ ($\mathbf{z} \in \mathbb{R}^D$) and then conduct a linear projection $\mathbf{z}^{proj} = \mathbf{z}\mathbf{E}_k$ to obtain the representation $\mathbf{z}^{proj} \in \mathbb{R}^K$, where $\mathbf{E}_k$ contains the first $K$ eigenvectors ranked by the corresponding eigenvalues (see Section 3). To reconstruct $\mathbf{x}$ from $\mathbf{z}^{proj}$, we firstly conduct an inverse projection $\hat{\mathbf{z}} = \mathbf{z}^{proj}\mathbf{E}_k^T$ and then obtain the reconstruction $\hat{\mathbf{x}}$ by letting $\hat{\mathbf{x}} = f(\hat{\mathbf{z}})$, where $f = g^{-1}$. In Figure 10 we show reconstructions with differing $K$ and may observe that good reconstruction can be obtained when $K$ is greater than 2700. The quality goes down as $K$ decreases, which empirically verifies the intrinsic dimension of the data distribution to be around $\sim$2700, which is consistent with our model estimation (as shown in Figure 9).

We also compare the sample quality of our model with the recently proposed CEF model Ross and Cresswell (2021), which proposes to learn a flow on manifold distributions. Distinct from our model, where different $K$ values can be selected post training, the CEF work requires an intrinsic model dimension to be pre-specified. We follow Ross and Cresswell (2021); select $K = 512$ and train the model on the CelebA ($32\times32$) dataset. Our model achieves better FID compared to CEF (see Table 5 in the Appendix for FID comparisons). The visualizations of the samples can be found in Figure 18 (our model with $K = \{2000, 3000\}$) and Appendix D (comparisons of both models). We also train our model on SVHN Netzer et al. (2011b) and CIFAR10 Krizhevsky et al. (2009a), see Appendix C for a detailed discussion.

## 6 Related Work

Classic latent variable generative models assume that data distributions lie *around* a low-dimensional manifold, for example, the Variational Auto-Encoder Kingma and Welling (2013) and the recently introduced Relaxed Injective Flow Kumar et al. (2020) (which uses an $L2$ reconstruction loss in the objective, implicitly assuming that the model has a Gaussian observation distribution, with isotropic variance). Such methods typically assume that observational noise is not degenerated (*e.g.* a fixed Gaus-



(a) $\texttt{Indim}(\mathbb{P}_d) = 5$ (b) $\texttt{Indim}(\mathbb{P}_d) = 10$ (c) Real MNIST

Figure 6: Eigenvalues estimated by our model trained on synthetic MNIST with $\texttt{Indim}(\mathbb{P}_d) = \{5, 10\}$ and real MNIST respectively.

sian distribution), therefore the model distribution is absolutely continuous and maximum likelihood learning is thus well defined Loaiza-Ganem et al. (2022); Zhang et al. (2020). However, common distributions such as natural images usually do not have Gaussian observational noise Zhao et al. (2017b). Therefore, we focus on modelling distributions that lie *on* a low-dimensional manifold.

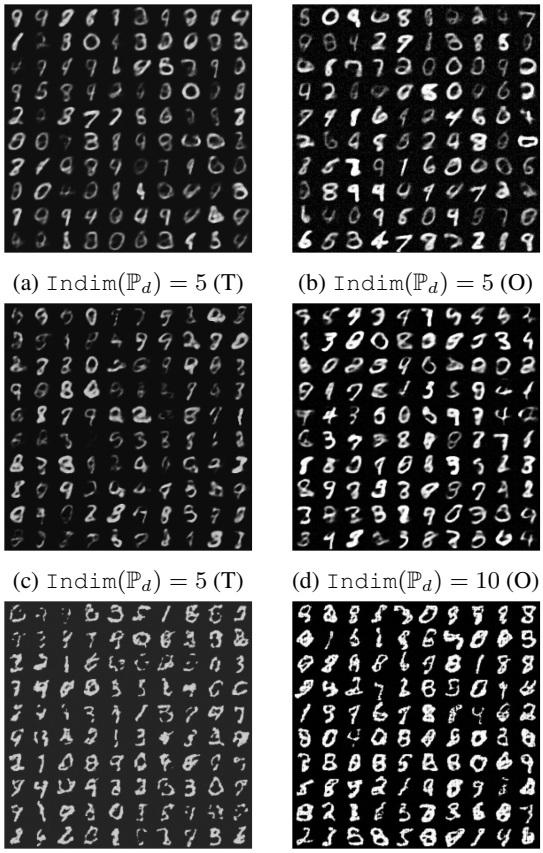
The study of manifold learning for nonlinear dimensionality reduction Cayton (2005) and intrinsic dimension estimation Camastra and Staiano (2016) is a rich field with an extensive set of tools. However, most methods commonly do not model the data density on the manifold and are thus not used for the same purpose as the models introduced in our work. There are however a number of recent works that introduce normalizing flows on manifolds that we now highlight and relate to our approach.

Several works define flows on manifolds with prescribed charts. Gemici et al. (2016) generalized flows from Euclidean spaces to Riemannian manifolds by proposing to map points from the manifold $\mathcal{M}$ to $\mathbb{R}^K$, apply a normalizing flow in this space and then mapping back to $\mathcal{M}$. The technique has since been further extended to Tori and Spheres Rezende et al. (2020). In contrast to our work, these methods require knowledge of the intrinsic dimension $K$ and a parameterization of the coordinate chart of the data manifold.

Several recently proposed manifold flow models can learn the data distribution without providing a chart mapping a priori. M-flow Brehmer and Cranmer (2020) proposed an algorithm that maps the data distribution to a lower-dimensional space with an auto-encoder and uses a flow in the lower-dimensional space to learn data distributions. Rectangular flow Caterini et al. (2021) and Conformal Embedding Flows Ross and Cresswell (2021) propose to directly build injective mappings from low-dimension space to high-dimension space. However, their methods still require that the dimensionality of the manifold is known. In Brehmer and Cranmer (2020), it is proposed that dimensionality can be learned either by a brute-force solution or through a trainable variance in the density function. The brute-force solution is clearly infeasible for data embedded in extremely high dimensional space, as is often encountered in deep learning tasks. Use of a trainable variance is natural and similar to our approach. However, as discussed at the beginning of this paper, without carefully handling the KL or MLE term in the objective, a vanishing variance parameter will result in the wild behaviour of the optimization process since these terms are not well defined.

The GIN model considered in Sorrenson et al. (2020) could recover the low-dimensional generating latent variables by following their identifiability theorem. However, the assumptions therein require knowledge of an auxiliary variable, *i.e*. the label, which is not required in our model. Behind this difference is the essential discrepancy between the concept of informative dimensions and intrinsic dimensions. The GIN model discovers the latent variables that are informative in a given context, defined by the auxiliary variable $u$ instead of the true intrinsic dimensions. In their synthetic example, the ten-dimensional data is a nonlinear transformation of ten-dimensional latent variables where two out of ten are correlated with the labels of the data and the other eight are not. In this example, there are two informative dimensions, but ten intrinsic dimensions. Nevertheless, our method for intrinsic dimension discovery can be used together with informative dimension discovery methods to discover finer structures of data.

In recent work Tempczyk et al. (2022), the authors made an intriguing observation regarding the impact of adding Gaussian noise $\mathcal{N}(0, \sigma^2 I)$ to a given data distribution. Specifically, they found that the change in log-likelihood, resulting from adding noise with different values of $\sigma$ is approximately linear in the logarithm of $\sigma$, with a proportionality constant that corresponds to the difference between the intrinsic and ambient dimensions of the data. Although both methods involve fitting flows to a noised data distribution, our method for estimating the intrinsic dimension is different: we estimate the intrinsic dimension using the rank of a learned degenerate Gaussian prior. We consider the future study of connections between these approaches may help to afford a deeper understanding of intrinsic dimensionalities with respect to complex data distributions.



(a) $\texttt{Indim}(\mathbb{P}_d) = 5$ (T) (b) $\texttt{Indim}(\mathbb{P}_d) = 5$ (O)

(c) $\texttt{Indim}(\mathbb{P}_d) = 5$ (T) (d) $\texttt{Indim}(\mathbb{P}_d) = 10$ (O)

(e) Real MNIST (T) (f) Real MNIST (O)

Figure 7: We compare the samples from a traditional non-volume preserving flow with a fixed prior (denoted as T) and our model (denoted as O) on synthetic MNIST with $\texttt{Indim}(\mathbb{P}_d) = \{5, 10\}$ and real MNIST, see Appendix C.5 for experiment details. We find that our model can provide improved sample quality on these manifold datasets.

## 7 Limitations and Future Work

In our work, we present a novel manifold flow model that utilizes a learnable generalized Gaussian prior to capture the low-dimensional manifold data distribution and identify their intrinsic dimensions. We demonstrated the benefits of our model in terms of sample generation and representation quality. While our model offers a promising step forward on the modelling of manifold distributions, it is important to note that we rely on a somewhat strong assumption: that the data distribution lies on a continuous manifold. This assumption may not hold for real-world image data, which typically exhibit complex and varied distributions. We conjecture that this limitation can be partially alleviated through the incorporation of prior techniques developed by Cornish et al. (2019); affording relaxation of the continuous manifold assumption and thus enable the flow model to capture data distributions that lie on a set of disconnected manifolds. We leave this to future work.

Figure 8: Samples from the proposed manifold prior with $K = 2000$ (left) and $K = 3000$ (right) respectively.

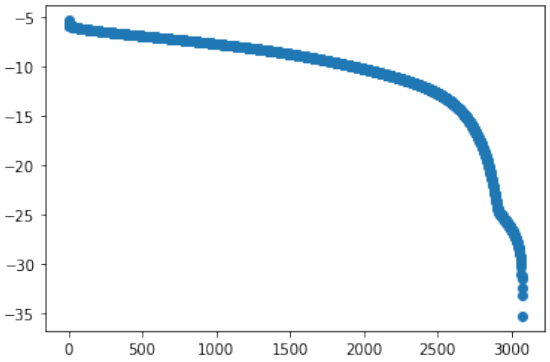
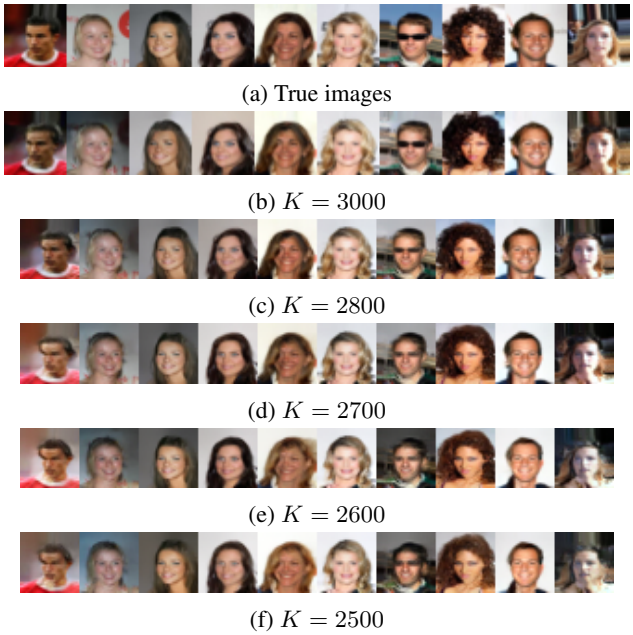



Figure 9: Eigen-values in log-space. We can see the eigenvalues has an exponential decay starting around 2700, which indicates that the intrinsic dimension of the CelebA (with size $32 \times 32$) is around 2700.



(a) True images

(b) $K = 3000$

(c) $K = 2800$

(d) $K = 2700$

(e) $K = 2600$

(f) $K = 2500$

Figure 10: Reconstructions with different dimensional representations. We can see the quality of the reconstruction slightly decreases when $K$ is lower than 2700, which empirically verified that the intrinsic dimension of the data distribution is around 2700.

## References


Syed Mumtaz Ali and Samuel D Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society: Series B (Methodological)*, 28(1):131–142, 1966.

M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. *arXiv:1701.04862*, 2017a.

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017b.

David Beymer and Tomaso Poggio. Image representations for visual learning. *Science*, 272(5270):1905–1909, 1996.

Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. *arXiv preprint arXiv:2003.13913*, 2020.

Francesco Camastra and Antonino Staiano. Intrinsic dimension estimation: Advances and open problems. *Information Sciences*, 328:26–41, 2016.

Anthony L Caterini, Gabriel Loaiza-Ganem, Geoff Pleiss, and John P Cunningham. Rectangular flows for manifold learning. *arXiv preprint arXiv:2106.01413*, 2021.

Lawrence Cayton. Algorithms for manifold learning. *Univ. of California at San Diego Tech. Rep*, 12(1-17):1, 2005.

Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images. *arXiv preprint arXiv:2011.10650*, 2020.

Rob Cornish, Anthony L Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. *arXiv preprint arXiv:1909.13833*, 2019.

Jose A Costa and Alfred O Hero. Determining intrinsic dimension and entropy of high-dimensional shape spaces. In *Statistics and Analysis of Shapes*, pages 231–252. Springer, 2006.

Bin Dai and David Wipf. Diagnosing and enhancing vae models. *arXiv preprint arXiv:1903.05789*, 2019.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):1–8, 2017.

Mevlana C Gemici, Danilo Rezende, and Shakir Mohamed. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*, 2016.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

Matthias Hein and Jean-Yves Audibert. Intrinsic dimensionality estimation of submanifolds in rd. In *Proceedings of the 22nd international conference on Machine learning*, pages 289–296, 2005.

Marco F Huber, Tim Bailey, Hugh Durrant-Whyte, and Uwe D Hanebeck. On entropy approximation for gaussian mixture random vectors. In *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 181–188. IEEE, 2008.

Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Ioannis Kontoyiannis and Mokshay Madiman. Sumset and inverse sumset inequalities for differential entropy and mutual information. *IEEE transactions on information theory*, 60(8):4503–4514, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009a.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009b.

Abhishek Kumar, Ben Poole, and Kevin Murphy. Regularized autoencoders via relaxed injective probability flow. In *International Conference on Artificial Intelligence and Statistics*, pages 4292–4301. PMLR, 2020.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

John M Lee. Smooth manifolds. In *Introduction to Smooth Manifolds*, pages 1–31. Springer, 2013.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015a.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015b.

Gabriel Loaiza-Ganem, Brendan Leigh Ross, Jesse C Cresswell, and Anthony L Caterini. Diagnosing and fixing manifold overfitting in deep generative models. *arXiv preprint arXiv:2204.07172*, 2022.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011a.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011b.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

Athanasios Papoulis and S Unnikrishna Pillai. *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, 2002.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Danilo Jimenez Rezende, George Papamakarios, Sébastien Racanière, Michael S Albergo, Gurtej Kanwar, Phiala E Shanahan, and Kyle Cranmer. Normalizing flows on tori and spheres. *arXiv preprint arXiv:2002.02428*, 2020.

Brendan Leigh Ross and Jesse C Cresswell. Tractable density estimation on learned manifolds with conformal embedding flows. *arXiv preprint arXiv:2106.05275*, 2021.

Paul K Rubenstein, Bernhard Schoelkopf, and Ilya Tolstikhin. On the latent space of wasserstein auto-encoders. *arXiv preprint arXiv:1802.03761*, 2018.

Peter Sorrenson, Carsten Rother, and Ullrich Kothe. Disentanglement by nonlinear ica with general incompressible-flow networks (gin). *arXiv preprint arXiv:2001.04872*, 2020.

Piotr Tempczyk, Rafał Michaluk, Lukasz Garncarek, Przemysław Spurek, Jacek Tabor, and Adam Golinski. Lidl: Local intrinsic dimension estimation using approximate likelihood. In *International Conference on Machine Learning*, pages 21205–21231. PMLR, 2022.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

Benigno Uria, Iain Murray, and Hugo Larochelle. Rnade: The real-valued neural autoregressive density-estimator. *arXiv preprint arXiv:1306.0186*, 2013.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

Raymond W Yeung. *Information theory and network coding*. Springer Science & Business Media, 2008.

Kai Yu and Tong Zhang. Improved local coordinate coding using local tangents. In *ICML*. Citeseer, 2010.

Mingtian Zhang, Thomas Bird, Raza Habib, Tianlin Xu, and David Barber. Variational f-divergence minimization. *arXiv preprint arXiv:1907.11891*, 2019.

Mingtian Zhang, Peter Hayes, Tom Bird, Raza Habib, and David Barber. Spread Divergences. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, 2020.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017a.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards deeper understanding of variational autoencoding models. *arXiv preprint arXiv:1702.08658*, 2017b.

# A Maximum Likelihood Estimation and KL divergence

Given data $x_1, x_2, \ldots, x_N$ sampled independently from the true data distribution $\mathbb{P}_d$, with density function $p_d(x)$, we want to fit the model density $p(x)$[7] to the data. A popular choice to achieve this involves minimization of the KL divergence where:

$$\mathrm{KL}(p_d(x)||p(x)) = \int p_d(x) \log p_d(x) dx - \int p_d(x) \log p(x) dx \tag{14}$$

$$= -\int p_d(x) \left( \log p_Z\left(g(x)\right) + \log \left| \det \left( \frac{\partial g}{\partial x} \right) \right| \right) dx + const.. \tag{15}$$

Since we can only access samples from $p_d(x)$, we approximate the integral by Monte Carlo sampling

$$\mathrm{KL}(p_d(x)||p(x)) \approx -\frac{1}{N} \sum_{n=1}^{N} \log p(x_n) + const.. \tag{16}$$

Therefore, minimizing the KL divergence between the data distribution and the model is (approximately) equivalent to Maximum Likelihood Estimation (MLE).

When $p(x)$ is a flow-based model with invertible flow function $f : Z \to X, g = f^{-1}$, minimizing the KL divergence in $X$ space is equivalent to minimizing the KL divergence in the $Z$ space. We let $X_d$ be the random variable of the data distribution and define $Z_{\mathbb{Q}} : Z_{\mathbb{Q}} = g(X_d)$ with density $q(z)$, so $q(z)$ can be represented as

$$q(z) = \int \delta(z - g(x)) p_d(x) dx. \tag{17}$$

Let $p(z)$ be the density of the prior distribution $\mathbb{P}_Z$, the KL divergence in $Z$ space can be written as

$$\mathrm{KL}(q(z)||p(z)) = \underbrace{\int q(z) \log q(z) dz}_{\text{Term 1}} - \underbrace{\int q(z) \log p(z) dz}_{\text{Term 2}}. \tag{18}$$

Term 1: using the properties of transformation of random variables Papoulis and Pillai (2002, pp. 660), the negative entropy can be written as

$$\int q(z) \log q(z) dz = \underbrace{\int p_d(x) \log p_d(x) dx}_{const.} - \int p_d(x) \log \left| \det \left( \frac{\partial g}{\partial x} \right) \right| dx. \tag{19}$$

Term2: the cross entropy can be written as

$$\int q(z) \log p(z) dz = \int \int \delta(z - g(x)) p_d(x) \log p(z) dz dx \tag{20}$$

$$= \int p_d(x) \log p(g(x)) dx. \tag{21}$$

Therefore, the KL divergence in $Z$ space is equivalent to the KL divergence in $X$ space

$$\mathrm{KL}(q(z)||p(z)) = \mathrm{KL}(p_d(x)||p(x)). \tag{22}$$

We thus build the connection between MLE and minimizing the KL divergence in $Z$ space.

# B Entropy

## B.1 An example

Assume a 2D Gaussian random variable $X$ with covariance $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. There exits two volume-perserving flows $g$ with parameters $\theta_1$ and $\theta_2$ ($\theta_1 \neq \theta_2$) to make $Z_1 = g_{\theta_1}(X)$ and $Z_2 = g_{\theta_2}(X)$ be Gaussians with covariance $\begin{bmatrix} 2 & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$ and

[7] For brevity, we use notation $p(x)$ to represent the model $p_X(x)$ unless otherwise specified.

$\begin{bmatrix}3 & 0\\ 0 & \frac{1}{3}\end{bmatrix}$ respectively. In this case, the entropy $\mathrm{H}(Z_1) = \mathrm{H}(Z_2) = \mathrm{H}(X)$ and does not depend on $\theta$. However, for a given Gaussian variable $K$, $\mathrm{H}(Z_1 + K) \neq \mathrm{H}(Z_2 + K)$ and $\mathrm{H}(g_\theta(X) + K)$ will depend on $\theta$. For example, the distribution of $K$ is a Gaussian with covariance $\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$, so $Z_1 + K$ is a Gaussian with covariance $\begin{bmatrix}3 & 0\\ 0 & \frac{3}{2}\end{bmatrix}$ and $Z_2 + K$ is a Gaussian with covariance $\begin{bmatrix}4 & 0\\ 0 & \frac{4}{3}\end{bmatrix}$, so $\mathrm{H}(Z_1 + K) \neq \mathrm{H}(Z_2 + K)$. A similar example can be constructed when $X$ is not absolutely continuous.

### B.2 Z is an absolutely continuous random variable

For two mutually independent absolutely continuous random variable $Z$ and $K$, the mutual information between $Z + K$ and $K$ is

$$\begin{aligned}\mathrm{I}(Z + K, K) &= \mathrm{H}(Z + K) - \mathrm{H}(Z + K|K) &(23)\\ &= \mathrm{H}(Z + K) - \mathrm{H}(Z|K) &(24)\\ &= \mathrm{H}(Z + K) - \mathrm{H}(Z). &(25)\end{aligned}$$

The last equality holds because $Z$ and $K$ are independent. Since mutual information $\mathrm{I}(Z + K, K) \geq 0$, we have

$$\mathrm{H}(Z) \leq \mathrm{H}(Z + K) = \mathrm{H}(Z) + \mathrm{I}(Z + K, K). \tag{26}$$

Assume $K$ has a Gaussian distribution with 0 mean and variance $\sigma_Z^2$[8]. When $\sigma_Z^2 = 0$, $K$ degenerates to a delta function, so $Z + K = Z$ and

$$\mathrm{I}(Z + K, K) = \mathrm{H}(Z + K) - \mathrm{H}(Z) = 0, \tag{27}$$

this is because the mutual information between an a.c. random variable and a singular random variable is still well defined, see Yeung (2008, Theorem 10.33). Assume $K_1$, $K_2$ are Gaussian random variables with 0 mean and variances $\sigma_1^2$ and $\sigma_2^2$ respectively. Without loss of generality, we assume $\sigma_1^2 > \sigma_2^2$ and $\sigma_1^2 = \sigma_2^2 + \sigma_\delta^2$, and let $K_\delta$ be the random variable of a Gaussian that has 0 mean and variance $\sigma_\delta^2$ such that $K_1 = K_2 + K_\delta$. By the data-processing inequality, we have

$$\mathrm{I}(Z + K_2, K_2) \leq \mathrm{I}(Z + K_2 + K_\delta, K_2 + K_\delta) = \mathrm{I}(Z + K_1, K_1). \tag{28}$$

Therefore, $\mathrm{I}(Z + K, K)$ is a monotonically decreasing function when $\sigma_Z^2$ decreases and when $\sigma_Z^2 \to 0$, $I(Z + K, K) \to 0$.

### B.3 Upper bound of the spread KL divergence

In this section, we show that leaving out the entropy term $\mathrm{H}(Z_{\tilde{\mathbb{Q}}})$ in equation 10 is equivalent to minimizing an *upper bound* of the spread KL divergence.

For singular random variable $Z_\mathbb{Q} = g(X_d)$ and absolutely continuous random variable $K$ that are independent, we have

$$\begin{aligned}\mathrm{H}(Z_\mathbb{Q} + K) - \mathrm{H}(K) &= \mathrm{H}(Z_\mathbb{Q} + K) - \mathrm{H}(Z_\mathbb{Q} + K|Z_\mathbb{Q}) &(29)\\ &= \mathrm{I}(Z_\mathbb{Q} + K, Z_\mathbb{Q}) \geq 0. &(30)\end{aligned}$$

The second equation is from the definition of Mutual Information (MI); the MI between an a.c. random variable and a singular random variable is well defined and always positive, see Yeung (2008, Theorem 10.33) for a proof.

Therefore, we can construct an upper bound of the spread KL objective in equation 10

$$\begin{aligned}\mathrm{KL}(\tilde{q}||\tilde{p}) &= \underbrace{\int q(\tilde{\mathbf{z}}) \log q(\tilde{\mathbf{z}}) d\tilde{\mathbf{z}}}_{-\mathrm{H}(Z_\mathbb{Q}+K)} - \int q(\tilde{\mathbf{z}}) \log p(\tilde{\mathbf{z}}) d\tilde{\mathbf{z}} &(31)\\ &= \underbrace{-\mathrm{H}(K)}_{const.} - \underbrace{\mathrm{I}(Z_\mathbb{Q} + K, Z_\mathbb{Q})}_{\geq 0} - \int q(\tilde{\mathbf{z}}) \log p(\tilde{\mathbf{z}}) d\tilde{\mathbf{z}}. &(32)\\ &\leq \underbrace{-\mathrm{H}(K)}_{const.} - \int q(\tilde{\mathbf{z}}) \log p(\tilde{\mathbf{z}}) d\tilde{\mathbf{z}}. &(33)\end{aligned}$$

[8] The extension to higher dimensions is straightforward.

Therefore, ignoring the negative entropy term during training is equivalent to minimizing an upper bound of the spread KL objective, and the gap between the bound and the true objective is $\mathrm{I}(Z_{\mathbb{Q}} + K, Z_{\mathbb{Q}})$. Unlike the case when $Z_{\mathbb{Q}}$ is a.c., $\mathrm{I}(Z_{\mathbb{Q}} + K, Z_{\mathbb{Q}})$ will be close to 0 when the variance of $K$ is sufficiently small, we do not have the same result when $Z_{\mathbb{Q}}$ is not absolutely continuous. However, we show that, under some assumptions of the flow function, the gap $\mathrm{I}(Z_{\mathbb{Q}} + K, Z_{\mathbb{Q}})$ can be upper bounded.

### B.4 Empirical evidence for ignoring the negative entropy

In this section, we first introduce the approximation technique to compute the negative entropy term, and then discuss the contribution of this term during training. The negative entropy of random variable $Z_{\tilde{Q}}$ is

$$-\mathrm{H}(Z_{\tilde{\mathbb{Q}}}) = \int q(\tilde{\mathbf{z}}) \log q(\tilde{\mathbf{z}}) d\tilde{\mathbf{z}}, \tag{34}$$

where

$$q(\tilde{\mathbf{z}}) = \int_{\mathbf{z}} p_K(\tilde{\mathbf{z}} - \mathbf{z}) d\mathbb{Q}_Z, \tag{35}$$

and $p_K$ is the density of a Gaussian with diagonal covariance $\sigma_Z^2 I$. We first approximate $\tilde{q}(\tilde{\mathbf{z}})$ by a mixture of Gaussians

$$q(\tilde{\mathbf{z}}) \approx \frac{1}{N} \sum_{n=1}^{N} \mathcal{N}(\tilde{\mathbf{z}}; \mathbf{z}^n, \sigma_Z^2 I) \equiv \hat{q}^N(\tilde{\mathbf{z}}) \tag{36}$$

where $\mathbf{z}^n$ is the $n$th sample from distribution $\mathbb{Q}_z$ by first sampling $\mathbf{x}^n \sim \mathbb{P}_d$ and letting $\mathbf{z}^n = g(\mathbf{x}^n)$. We denote the random variable of this Gaussian mixture as $\hat{Z}^N_{\tilde{\mathbb{Q}}}$ and approximate

$$-\mathrm{H}(Z_{\tilde{\mathbb{Q}}}) \approx -\mathrm{H}(\hat{Z}^N_{\tilde{\mathbb{Q}}}). \tag{37}$$

However, the entropy of a Gaussian mixture distribution does not have a closed form, so we further conduct a first order Taylor expansion approximation Huber et al. (2008)

$$-\mathrm{H}(\hat{Z}^N_{\tilde{\mathbb{Q}}}) \approx \frac{1}{N} \sum_{n=1}^{N} \log \hat{q}^N(\tilde{\mathbf{z}} = \mathbf{z}^n), \tag{38}$$

this approximation is accurate for small $\sigma_Z^2$. Finally we have our approximation

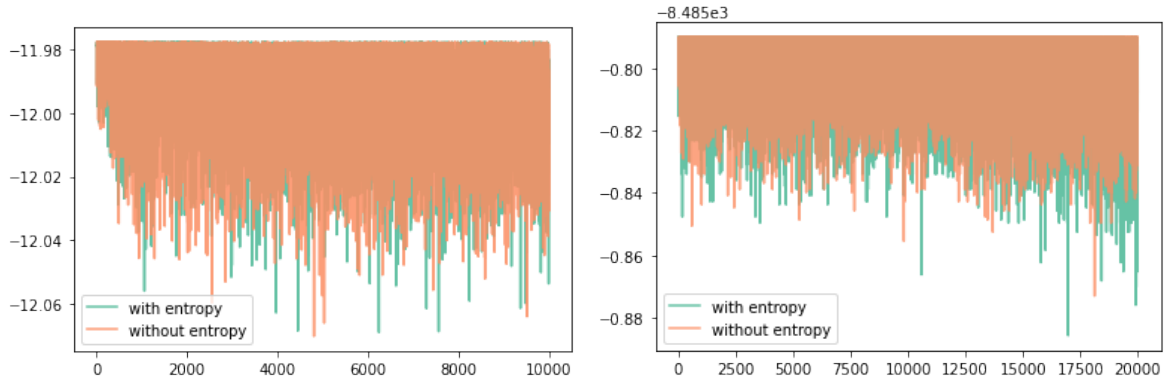
$$-\mathrm{H}(Z_{\tilde{\mathbb{Q}}}) \approx \frac{1}{N} \sum_{n=1}^{N} \log \hat{q}^N(\tilde{\mathbf{z}} = \mathbf{z}^n). \tag{39}$$

To evaluate the contribution of the negative entropy, we train our flow model on both low dimensional data (Toy datasets: 2D) and high dimensional data (Fading square dataset: 1024D) by optimization that uses two objectives: (1) ignoring the negative entropy term in equation 7 and (2) approximating the negative entropy term in equation 7 using the approximation discussed above. During training, we keep track of the value of the entropy $\mathrm{H}(Z_{\tilde{\mathbb{Q}}})$ (using the approximation value) in both objectives. We let $N$ equal the batch size when approximating the entropy. Additional training details remain consistent with those described in Appendix C.

In Figure 11, we plot the (approximated) entropy value $\mathrm{H}(Z_{\tilde{\mathbb{Q}}})$ during training for both experiments. We find the difference between having the (approximated) negative entropy term and ignoring the negative entropy to be negligible. We leave rigorous theoretical investigation on the effects of discarding the entropy term during training to future work.

## C Experiments

We conduct all our experiments on one single NVDIA GeForce RTX 2080 Ti GPU.




(a) Toy dataset

(b) Fading square dataset

Figure 11: Figure (a) and (b) show the (approximated) entropy value using two different training objectives, for two different experiments.

### C.1 Network architecture for toy and MNIST dataset

The flow network we use consists of incompressible affine coupling layers Sorrenson et al. (2020); Dinh et al. (2016). Each coupling layer splits a $D$-dimensional input $\mathbf{x}$ into two parts $\mathbf{x}_{1:d}$ and $\mathbf{x}_{d+1:D}$. The output of the coupling layer is

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d} \tag{40}$$

$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d}), \tag{41}$$

where $s : \mathbb{R}^d \to \mathbb{R}^{D-d}$ and $t : \mathbb{R}^d \to \mathbb{R}^{D-d}$ are scale and translation functions parameterized by neural networks, $\odot$ is the element-wise product. The log-determinant of the Jacobian of a coupling layer is the sum of the scaling function $\sum_j s(\mathbf{x}_{1:d})_j$. To make the coupling transform volume preserving, we normalize the output of the scale function, so the $i$-th dimension of the output is

$$\tilde{s}(\mathbf{x}_{1:d})_i = s(\mathbf{x}_{1:d})_i - \frac{1}{D-d}\sum_j s(\mathbf{x}_{1:d})_j, \tag{42}$$

and the log-determinant of the Jacobian is $\sum_i \tilde{s}(\mathbf{x}_{1:d})_i = 0$. We compare a volume-preserving flow with a learnable prior (normalized $s(\cdot)$) to a non-volume preserving flow with a fixed prior (unnormalized $s(\cdot)$). In this way both models have the ability to adapt their 'volume' to fit the target distribution, retaining comparison fairness.

In our affine coupling layer, the scale function $s$ and the translation function $t$ have two types of structure: fully connected net and convolution net. Each fully connected network is a 4-layer neural network with hidden-size 24 and Leaky ReLU with slope 0.2. Each convolution net is a 3-layer convolutional neural network with hidden channel size 16, kernel size 3×3 and padding size 1. The activation is Leaky ReLU with a slope 0.2. The downsampling decreases the image width and height by a factor of 2 and increases the number of channels by 4 in a checkerboard-like fashion Sorrenson et al. (2020); Jacobsen et al. (2018). When multiple convolutional nets are connected together, we randomly permute the channels of the output of each network except the final one. In Table 1, 2, 3, 4, we show the network structures for our four main paper experiments.

### C.2 Toy dataset

We also construct a second dataset and train a flow model using the same training procedure discussed in Section 5. Figure 12a shows the samples from the data distribution $\mathbb{P}_d$. Each data point is a 2D vector $\mathbf{x} = [x_1, x_2]$ where $x_1 \sim \mathcal{N}(0, 1)$ and $x_2 = x_1$, so $\texttt{Indim}(\mathbb{P}_d) = 1$. Figure 12f shows that the prior $\mathbb{P}_Z$ has learned the true intrinsic dimension $\texttt{Indim}(\mathbb{P}_Z) = \texttt{Indim}(\mathbb{P}_d) = 1$. We train our model using learning rate $3 \times 10^{-4}$ and batch size 100 for $10k$ iterations. We compare to samples drawn from a flow model that uses a fixed 2D Gaussian prior, with results shown in Figure 12. We can observe, for this simple dataset, flow with a fix Gaussian prior can generate reasonable samples, but the 'curling up' behavior, discussed in the main paper, remains highly evident in the $Z$ space, see Figure 12c.

**Manifold density** We also plot the 'density allocation' on the manifold for the two toy datasets. For example, for the data generation process $\mathbf{x} = [x_1, x_2]$ where $x_1 \sim p = \mathcal{N}(0, 1)$ and $x_2 = x_1$, we use the density value $p(x = x_1)$ to indicate

the 'density allocation' on the 1D manifold. To plot the 'density allocation' of our learned model, we first sample $\mathbf{x}_s$ uniformly from the support of the data distribution, the subscript 's' here means that they only contain the information of the *support*. Specifically, since $\mathbf{x}_s = [x_1^s, x_2^s]$, we sample $x_1^s \sim p = \mathcal{U}(-3\sigma, 3\sigma)$ ($\sigma$ is the standard deviation of $\mathcal{N}(0,1)$, $\mathcal{U}$ is the uniform distribution) and let $x_2^s = x_1^s$ or $x_2^s = \sin(x_1^s)$, depending on which dataset is used. We use the projection procedure that was described in Section 5 to obtain the projected samples $\mathbf{z}_s^{proj}$, so $\mathbf{z}_s^{proj} \in \mathbb{R}^{\texttt{Indim}(\mathbb{P}_d)}$. We also project the learned prior $\mathbb{P}_Z$ to $\mathbb{R}^{\texttt{Indim}(\mathbb{P}_d)}$ by constructing $\mathbb{P}_Z^{proj}$ as a Gaussian with zero mean and a diagonal covariance contains the non-zeros eigenvalues of $AA^T$. Therefore $\mathbb{P}_Z^{proj}$ is a.c. in $\mathbb{R}^{\texttt{Indim}(\mathbb{P}_d)}$ and we denote its density function as $p^{proj}(\mathbf{z})$. We then use the density value $p^{proj}(\mathbf{z} = \mathbf{z}_s^{proj})$ to indicate the 'density allocation' at the location of $\mathbf{x}_s$ on the manifold support. In Figure 13, we compare our model with the ground truth and find that we can successfully capture the manifold density.

| Type of block | Number | Input shape | Affine coupling layer widths |
|---|---|---|---|
| Fully connected | 2 | 2 | $1 \to 24 \to 24 \to 24 \to 1$ |

Table 1: The network structure for toy datasets.

| Type of block | Number | Input shape | Affine coupling layer widths |
|---|---|---|---|
| § Fully connected | 6 | 3 | even: $2 \to 24 \to 24 \to 24 \to 1$<br>odd: $1 \to 24 \to 24 \to 24 \to 2$ |

Table 2: The network structure for S-Curve dataset. If we denote the input vector of each coupling is $\mathbf{x}$, the function $s$ and $t$ takes $\mathbf{x}[0]$ in the first coupling layer and $\mathbf{x}[1]$ in the second coupling layer.

| Type of block | Number | Input shape | Affine coupling layer widths |
|---|---|---|---|
| Downsampling | 1 | $(1, 32, 32)$ | |
| Convolution | 2 | $(4, 16, 16)$ | $2 \to 16 \to 16 \to 4$ |
| Downsampling | 1 | $(4, 16, 16)$ | |
| Convolution | 2 | $(16, 8, 8)$ | $8 \to 16 \to 16 \to 16$ |
| Flattening | 1 | $(16, 8, 8)$ | |
| Fully connected | 2 | 1024 | $512 \to 24 \to 24 \to 24 \to 512$ |

Table 3: The network structure for our fading square dataset experiments. If we denote the input vector of each coupling to be $\mathbf{x}$, the functions $s$ and $t$ take values $\mathbf{x}[1{:}2]$ in the even coupling layer and $\mathbf{x}[3]$ in the odd coupling layer.

| Type of block | Number | Input shape | Affine coupling layer widths |
|---|---|---|---|
| Downsampling | 1 | $(1, 28, 28)$ | |
| Convolution | 4 | $(4, 14, 14)$ | $2 \to 16 \to 16 \to 4$ |
| Downsampling | 1 | $(4, 14, 14)$ | |
| Convolution | 4 | $(16, 7, 7)$ | $8 \to 16 \to 16 \to 16$ |
| Flattening | 1 | $(16, 7, 7)$ | |
| Fully connected | 2 | 784 | $392 \to 24 \to 24 \to 24 \to 392$ |

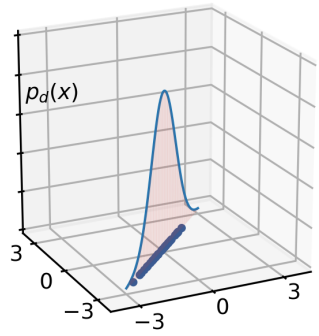
Table 4: Network structure for synthetic MNIST dataset experiment.

### C.3 S-Curve dataset

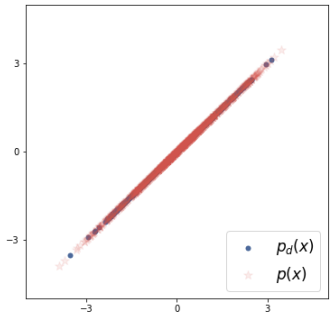
To fit our model to the data, we use the Adam optimizer with learning rate $5 \times 10^{-4}$, batch size $500$ and train the model for $200k$ iterations. We anneal the learning rate with a factor of $0.9$ every $10k$ iteration.

### C.4 Fading Square dataset

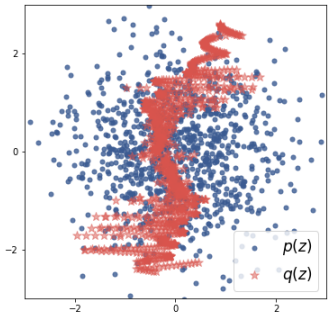
To fit the data, we train our model for $20k$ iterations with a batch size of $100$ using the Adam optimizer. The learning rate is initialized to $5{\times}10^{-4}$ and decays with a factor of $0.9$ at every $1k$ iteration. We additionally use an $L2$ weight decay with factor $0.1$.



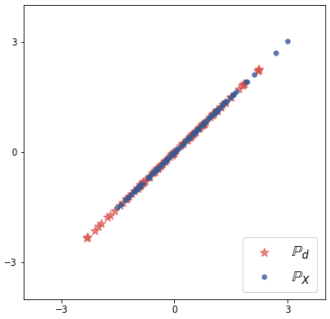

(a) Data distribution



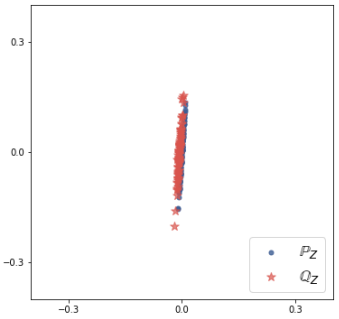

(b) Fixed Gaussian prior: $X$ space



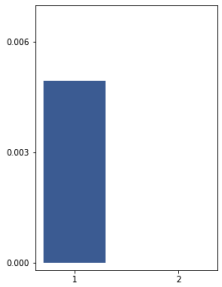

(c) Fixed Gaussian prior: $Z$ space




(d) Our method: $X$ space




(e) Our method: $Z$ space




(f) Eigenvalues of $AA^T$

Figure 12: (a) shows the samples from the data distribution. (b) and (d) show samples from a flow with a fixed Gaussian prior and our method, (c) and (e) show the latent space in both models. In (f), we plot the eigenvalues of $AA^T$.

### C.5 MNIST Dataset

#### C.5.1 Implicit data generation model

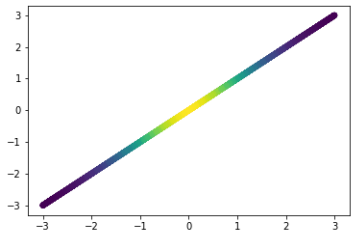
To fit an implicit model to the MNIST dataset, we first train a Variational Auto-Encoder (VAE) Kingma and Welling (2013) with Gaussian prior $p(\mathbf{z})$=$\mathcal{N}(0, I)$. The encoder is $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu_\theta(\mathbf{x}), \Sigma_\theta(\mathbf{x}))$ where $\Sigma$ is a diagonal matrix. Both $\mu_\theta$ and $\Sigma_\theta$ are parameterized by a 3-layer feed-forward neural network with a ReLU activation and the size of the two hidden outputs are 400 and 200. We use a Gaussian decoder $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(g_\theta(\mathbf{z}), \sigma_\mathbf{x}^2 I)$ with fixed variance $\sigma_\mathbf{x} = 0.3$. The $g_\theta$ is parameterized by a 3-layer feed-forward neural network with hidden layer sizes 200 and 400. The activation of the hidden output uses a ReLU and we utilize a Sigmoid function in the final layer output to constrain the output between 0 and 1. The training objective is to maximize the lower bound of the log-likelihood

$$\log p(\mathbf{x}) \geq \int q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{x}|\mathbf{z}) dz - \mathrm{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})),$$

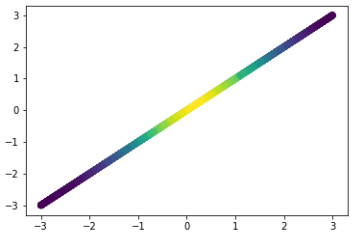
see Kingma and Welling (2013) for further details. We use an Adam optimizer with a learning rate $1 \times 10^{-4}$ and batch size 100 to train the model for 100 epochs. After training, we sample from the model by first taking a sample $\mathbf{z} \sim p(\mathbf{z})$ and letting $\mathbf{x} = g_\theta(\mathbf{z})$. This is equivalent to taking a sample from an implicit model $p_\theta(\mathbf{x}) = \int \delta(\mathbf{x} - g_\theta(\mathbf{z}))d(\mathbf{z})d\mathbf{z}$, see Tolstikhin et al. (2017) for further discussion regarding this implicit model construction. In Figure 14, we plot samples from the trained implicit model with $\texttt{dim}(\mathbf{z}) = 5$, $\texttt{dim}(\mathbf{z}) = 10$ and the original MNIST data.

#### C.5.2 Flow model training

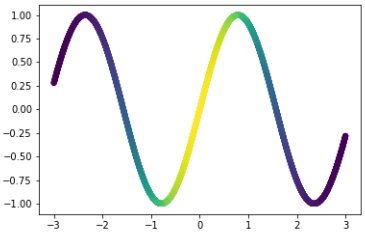
We train our flow models to fit the synthetic MNIST dataset with intrinsic dimensions 5 and 10 and the original MNIST dataset. In all models, we use the Adam optimizer with learning rate $5 \times 10^{-4}$ and batch size 100. We train the model for $300k$ iterations and, following the initial $100k$ iterations, the learning rate decays every $10k$ iterations by a factor 0.9.



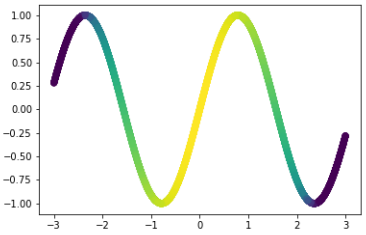
(a) Ground truth



(b) Our method



(c) Ground truth



(d) Our method

Figure 13: (a) and (c) show the ground truth 'density allocation' on the manifold for two toy datasets, (b) and (d) show the 'density allocation' learned by our models.

In Figure 15, we plot the samples from our models trained on three different training datasets. In Figure 16, we also plot the samples from traditional flow models that were trained on these three datasets using the same experiment settings. We find that the samples from our models are comparably sharper than those from traditional flow models.

### C.6 Color image experiments

Our network for color image datasets is based on an open source implementation[9], where we modify the Glow Kingma and Dhariwal (2018) structure to make it volume preserving. The volume preserving affine coupling layer is similar to the incompressible flow, however here the $s(\cdot)$ function takes the form:

$$s(\cdot) = \text{scale} * \tanh(\text{NN}(\cdot)) \tag{43}$$

where 'scale' are channel-wise learnable parameters and $\text{NN}(\cdot)$ is the neural network. For the actnorm function $\mathbf{y}_{ij} = \mathbf{s} \odot \mathbf{x}_{ij} + \mathbf{b}$, where the log-determinant for position $\mathbf{x}_{ij}$ is $\text{sum}(\log|\mathbf{s}|)$, we normalize the $\widetilde{\log|\mathbf{s}|} = \log|\mathbf{s}| - \text{mean}(\log|\mathbf{s}|)$, such that $\text{sum}(\widetilde{\log|\mathbf{s}|}) = 0$. We use LU decomposition for the $1\times1$ convolution, where $W = PL(U + \text{diag}(\mathbf{s}))$ with log-determinant $\log|\det(W)| = \text{sum}(\log|\mathbf{s}|)$. We can then normalize $\widetilde{\log|\mathbf{s}|} = \log|\mathbf{s}| - \text{mean}(\log|\mathbf{s}|)$ to ensure it is volume preserving. All experiments utilise a Glow architecture consisting of four blocks of twenty affine coupling layers, applying the multi-scale architecture between each block. The neural network in each affine coupling layer contains three convolution layers with kernel sizes 3,1,3 respectively and ReLU activation functions. We use 512 channels for all hidden convolutional layers, the Adam optimizer with learning rate $1e^{-4}$ and a batch-size of 100, for all of our experiments.

We conduct further experiments on real-world color images: SVHN, CIFAR10 and CelebA. Data samples from SVHN and CIFAR10 are represented by $32\times32\times3$ dimension vectors and for CelebA our pre-processing involves centre-cropping each image and resizing to $32\times32\times3$. Pixels take discrete values from $[0, \ldots, 255]$ and we follow a standard dequantization process Uria et al. (2013): $x = \frac{x+\text{Uniform}(0,1)}{256}$ to transform each pixel to a continuous value in $[0, 1]$. The manifold structure of these image datasets has been less well studied and may be considered more complex than the datasets previously explored in this work; each potentially lying on a union of disconnected and discontinuous manifolds. Similar to

[9] https://github.com/chrischute/glow

Table 5: FID Comparisons of Samples. We report the FID values (lower indicates better quality) of our model with $K = [512, 2000, 2500, 3000]$. We surprisingly find that when we increase the dimension of the manifold, the FID becomes higher. We hypothesize that the manifold spanned by the leading eigenvectors can capture the images lying in the modes, resulting in a better FID value. We also compare our method to CEF Ross and Cresswell (2021) model with manifold dimension $K = 512$ and find our method can consistently achieve better FID value.

| Model | CEF (512) | Ours (512) | Ours (2000) | Ours (2500) | Ours (3000) |
|---|---|---|---|---|---|
| FID | 105.1 | **44.8** | 54.4 | 62.8 | 63.1 |

the MNIST experiment, we add small Gaussian noise (with standard deviation $\sigma_x$=0.01) for twenty initial training epochs in order to smooth the data manifold and then anneal the noise with a factor of $0.9$ for a further twenty epochs. Figure 17 provides samples from the models that are trained on these three datasets. We find that our models can generate realistic samples and thus provide evidence towards the claim that our volume-preserving flow, with learnable prior model, does not limit the expressive power of the network. Experimentally, in comparison with the Glow model, we do not observe consistent sample quality improvement and conjecture that the intrinsic dimension mismatch problem, pertaining to these color image datasets, is less severe than the previously considered grey-scale images and constructed toy data. The data manifold has large intrinsic dimensionality due to high redundancy and stochasticity in the RGB channels Yu and Zhang (2010). We then conjecture that color image data distributions typically have a relatively large number of intrinsic dimensions. This conjecture is consistent with recent generative model training trends: a latent variable model *e.g*. VAE usually requires very large latent dimension to generate high-quality images Vahdat and Kautz (2020); Child (2020). For example, in the state-of-the-art image generation work[10] Vahdat and Kautz (2020), a VAE with latent size $153600$ is used to fit a CIFAR10 dataset with shape $32\times32\times3$. These considerations lead us to believe that the estimation of intrinsic dimension for natural images forms an important future research direction. Towards tackling this, one proposal would involve using the results obtained from our model to inform the latent dimension of the latent variable models. We leave deeper study of the geometry of image manifolds to future work.

## D Empirical Verifications and Comparisons

We compare our model with the recently proposed CEF model Ross and Cresswell (2021), which is also proposed to learn a flow on manifold distribution. Different from our model where different $K$ values can be selected after training the model, in the CEF model, the intrinsic dimension of the model has to be pre-specified. We then choose $K = 512$ and train the model on the CelebA ($32 \times 32$) dataset. We reuse their open-source code `https://github.com/layer6ai-labs/CEF` and the $32 \times 32$ model architecture (see paper Ross and Cresswell (2021) for details). In Figure5, we compare the reconstructions and the samples from both models with the same manifold dimension $K = 512$. We can find the CEF model has a better reconstruction quality than our models, we hypothesize that this is because our model is not trained with an intrinsic dimension $K = 512$, so the reconstruction quality cannot be guaranteed when we use the latent code with dimension 512. However, we observe our model achieves better sample quality (see Table5 for a FID comparison) compared to CEF. This phenomenon (good reconstruction but bad sample quality) is similar to the latent space mismatch problem in the latent variable models Dai and Wipf (2019); Zhao et al. (2017a), where the aggregate distribution $q_\phi(z) = \int \delta(z - g(x))\mathbb{P}_d(x)$ mismatches the prior $p(z)$, ($g$ is the inverse flow function). Since our model can learn the prior and the manifold prior with $K = 512$ is constructed by the leading eigen-vectors, it can alleviate the prior mismatch problem and has better sample quality.

## E License

The CelebA Liu et al. (2015b) and SVHN Netzer et al. (2011a) dataset is available for non-commercial research purposes only. We didn't find licenses for CIFAR Krizhevsky et al. (2009b) and MNIST LeCun (1998). The CelebA dataset may contain personal identifications.

The code `https://github.com/chrischute/glow` we made use of is under the MIT License.

[10]The latent dimension in GAN based models Goodfellow et al. (2014); Arjovsky and Bottou (2017a); Gulrajani et al. (2017) are chosen to be small (*e.g*. latent variable has dimension 128 for CIFAR10), but the mode-collapse phenomenon, commonly observed in GANs, indicates that those with small latent dimension will underestimate the support of the data distribution.

(a) $\texttt{dim}(\mathbf{z}) = 5$ (b) $\texttt{dim}(\mathbf{z}) = 10$ (c) MNIST Dataset

Figure 14: Training data for the flow model. Figure (a) and (b) are synthetic MNIST samples from two implicit models with latent dimension 5 and 10 respectively. Figure (c) are samples from the original MNIST dataset.

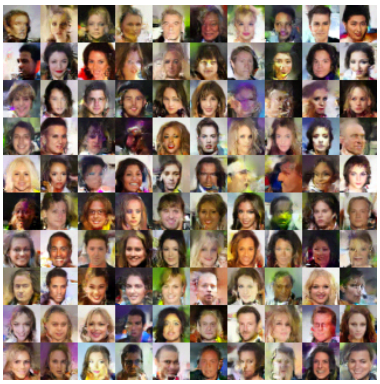
(a) $\texttt{dim}(\mathbf{z}) = 5$ (b) $\texttt{dim}(\mathbf{z}) = 10$ (c) MNIST

Figure 15: Samples from our methods. Figure (a) and (b) are samples flow models trained on synthetic MNIST data with intrinsic dimension 5 and 10 respectively. Figure (c) are samples from a flow model that trained on original MNIST data.

(a) $\texttt{dim}(\mathbf{z}) = 5$ (b) $\texttt{dim}(\mathbf{z}) = 10$ (c) MNIST

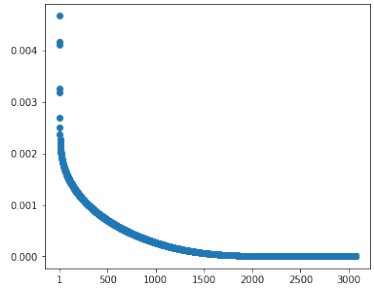
Figure 16: Samples from traditional non-volume preserving flow models with fixed Gaussian prior.

(a) SVHN Samples

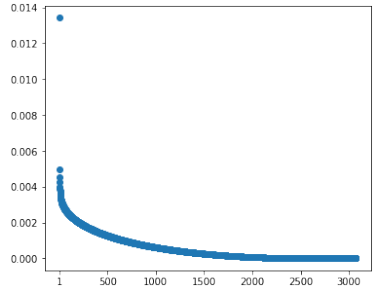


(b) CelebA Samples

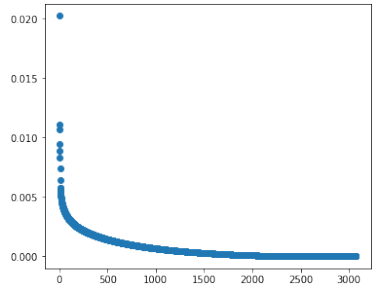
(c) CIFAR Samples




(d) SVHN Eigenvalues




(e) CelebA Eigenvalues

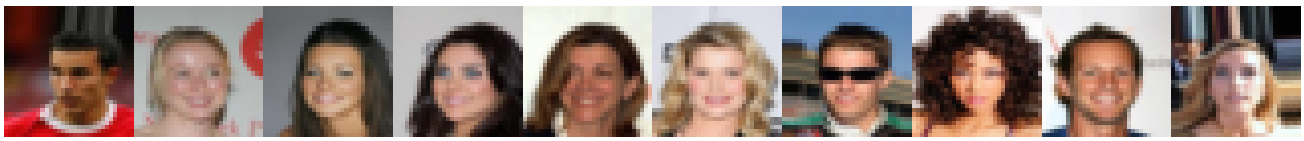



(f) CIFAR Eigenvalues

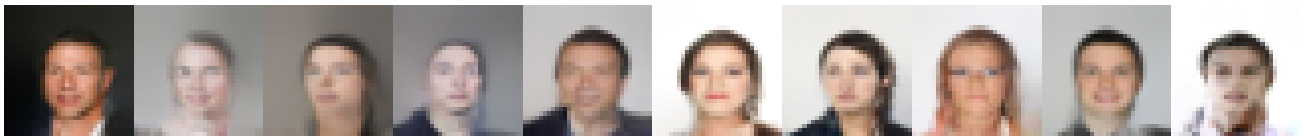
Figure 17: Samples and eigenvalues for three models. In each case we can find approximately half of the eigenvalues converge to 0 but a large number of eigenvalues remain greater than 0.

(a) Samples with $K = 1000$ (b) Samples with $K = 2000$ (c) Samples with $K = 2500$ (d) Samples with $K = 3000$

Figure 18: Samples from the model $K$-dimensional manifolds prior.



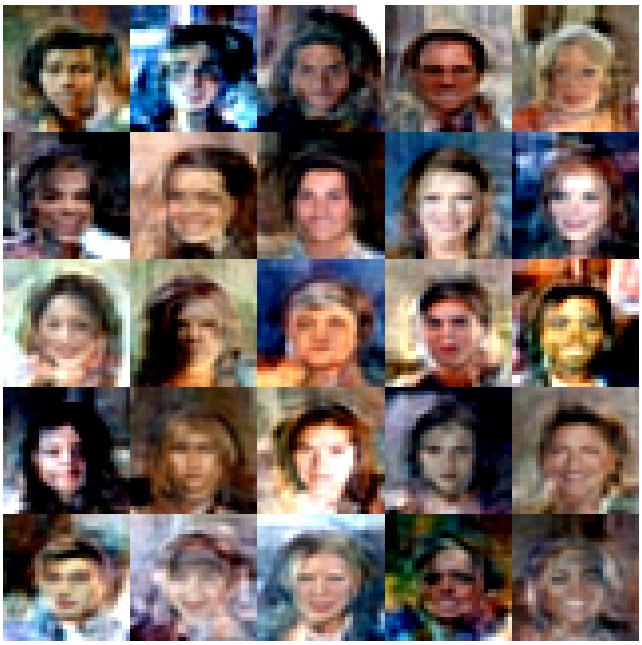
(a) Reconstruction Ground Truth



(b) Reconstruction from our model with $K = 512$

(c) Reconstruction from CEF with $K = 512$

(d) Samples from our model with $K = 512$



(e) Samples from CEF with $K = 512$

Figure 19: Comparisons with CEF model. We compare the reconstructions (a,b,c) and samples (d,c) from our model and CEF model with the same manifold dimension ($K = 512$), we can see CEF has a better reconstruction quality whereas our model has a better sample quality, also see 5 for FID comparisons and main text for a discussion of the phenomenon.