Flow Based Models For Manifold Data

Mingtian Zhang 1,2∗ Yitong Sun 2 Steven McDonagh 2 Chen Zhang 2

1AI Center, University College London mingtian.zhang.17@ucl.ac.uk
2Huawei Noah’s Ark Lab {sunyitong;steven.mcdonagh;chenzhang10}@huawei.com

Abstract

Flow-based generative 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 density will always have support off the data manifold, potentially resulting in degradation of model performance. In addition, the requirement for equal latent and data space dimensionality can unnecessarily increase complexity for contemporary flow models. Towards addressing these problems, we propose to learn a manifold prior that affords benefits to both sample generation and representation quality. An auxiliary benefit of our approach is the ability to identify the intrinsic dimension of the data distribution.

1 Introduction

Normalizing flows [35, 22] have shown considerable potential for the task of modelling and inferring expressive distributions through the learning of well-specified probabilistic models. Specifically, assume an absolutely continuous (a.c.) random variable (r.v.) Z with distribution P_Z and probability density p_Z(z). We can transform Z to get a r.v. X: X = f(Z), where f : R^D → 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

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

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 make use of MLE. We assume the data random variable X_d with distribution 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 KL(p_d(x)||p_X(x)) in X space (see Appendix A for detail), MLE is also equivalent to minimizing the KL divergence in Z space, due to the KL divergence invariance under invertible transformations [42, 33]. Specifically, we define Z_Q : Z_Q = g(X_d) with distribution Q_Z and density function

∗The work was done during an internship in Huawei Noah’s Ark Lab.

The distribution is absolutely continuous with respect to the Lebesgue measure. In this case, it has a density function, see [12].

Preprint. Under review.
The KL divergence in $Z$ space can be written as

$$KL(q(z)||p(z)) = \int q(z) \log q(z) dz - \int q(z) \log p(z) dz$$

$$= - \int p_d(x) \left( \log p_Z(g(x)) + \log \left| \det \frac{\partial g}{\partial x} \right| \right) dx + \text{const.}.$$  \hspace{1cm} (2)

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

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

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 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, MLE or the KL divergence 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 with manifold data

We assume a data sample $x \sim \mathbb{P}_d$ to be a $D$ dimensional vector $x \in \mathbb{R}^D$ and define the ambient dimensionality of $\mathbb{P}_d$, denoted by $\dim(\mathbb{P}_d)$, 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 [4]. We assume the dimensionality of the manifold to be $K$ where $K < D$, and define the intrinsic dimension of $\mathbb{P}_d$, denoted by $\indim(\mathbb{P}_d)$, 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 $x \sim \mathbb{P}_d$ is a 2D vector $x = \{x_1, x_2\}$ where $x_1 \sim \mathcal{N}(0, 1)$ and $x_2 = \sin(2x_1)$. Therefore, this example results in $\dim(\mathbb{P}_d) = 2$ and $\indim(\mathbb{P}_d) = 1$.

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 $\dim(\mathbb{P}_X) = \indim(\mathbb{P}_X) = D$. 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; $\indim(\mathbb{P}_X) \neq \indim(\mathbb{P}_d)$. 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 [20]. 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 dimension.

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 $p(z) = \mathcal{N}(0, I_2)$ and the function $f$ is a non-volume preserving flow with two coupling layers (see Appendix D.1). In Figure 1b we plot samples from the flow model; the sample $x$ is generated by first sampling a 2D datapoint $z \sim \mathcal{N}(0, I_2)$ and then letting $x = f(z)$. Figure 1c shows samples from the prior distributions $\mathbb{P}_Z$ and $\mathbb{Q}_Z$. $\mathbb{Q}_Z$ is defined as the transformation of $\mathbb{P}_Z$ using the bijective function $g$, such that $\mathbb{Q}_Z$ is constrained to support a 1D manifold in 2D space, and $\indim(\mathbb{Q}_Z) = \indim(\mathbb{P}_d) = 1$. Training of $\mathbb{Q}_Z$ to match $\mathbb{P}_Z$ (which has intrinsic dimension 2), can be seen in Figure 1c to result in curling up of the manifold in the latent space, contorting it towards satisfying a distribution that has intrinsic dimension 2. This ill-behaved phenomenon causes several potential problems for contemporary flow models:

\footnote{Since we assume $X_d$ is a.c. $Z_Q : Z_Q = g(X_d)$ is also a.c. with a bijection mapping $g(\cdot)$, thus $Z_Q$ allows a density function.}
Figure 1: Samples and latent visualization from a flow based model with a fixed Gaussian prior when the intrinsic dimension is strictly lower than the true dimensionality of the data space.

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 degradations of the representation quality.

3. **Inefficient use of network capacity.** Neural network capacity is spent on contorting the distribution \( Q_Z \) to satisfy imposed dimensionality constraints.

A natural solution to the problem of intrinsic dimension mismatch is to select a prior distribution \( P_Z \) with the same dimensionality as the intrinsic dimension of the data distribution such that:

\[
\text{Indim}(P_Z) = \text{Indim}(P_d).
\]

However, since we do not know \( \text{Indim}(P_d) \) explicitly, one option involves to instead learn it from the data distribution. In the following section, we will introduce a parameterization approach that enables us to learn \( \text{Indim}(P_d) \).

### 3 Learning a manifold prior

Consider a data vector \( x \in \mathbb{R}^D \), then a flow based model prior \( P_Z \) 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 \( \text{Indim}(P_Z) = D \).

To allow a prior to have intrinsic dimension strictly less than \( D \), we let \( P_Z \) have the “generalized density” \( p(z) \) with the form of a Generalized Gaussian distribution

\[
p(z) = \mathcal{G}N(0, AA^T), \tag{5}
\]

where \( z \in \mathbb{R}^D \) 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 \( P_Z \) is a (non-degenerate) multivariate Gaussian on \( \mathbb{R}^D \). When \( \text{Rank}(AA^T) = K \) and \( K < D \), then \( P_Z \) will degenerate to a Gaussian supported on a \( K \)-dimensional manifold, such that the intrinsic dimension \( \text{Indim}(P_Z) = K \). 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 \( P_Z \) is initialized as a standard Gaussian.

When \( \text{Rank}(AA^T) < D \), the degenerate covariance \( AA^T \) is no longer invertible and we are unable to evaluate the density value of \( p(z) \) for a given random vector \( z \). Furthermore, when the data distribution \( P_d \) is supported on a \( K \)-dimensional manifold, \( Q_Z \) 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 \( P_Z \) and \( Q_Z \) is not well defined5.

---

4We use the generalized density to include the case that \( AA^T \) is not full rank.

5The KL divergence \( \text{KL}(Q||P) \) is well defined when \( Q \) and \( P \) have valid densities and their densities have the same support [1].
Recent work by [45, 44] proposed a new family of divergence to address this problem. In the following section we briefly review key concepts pertaining to this family of divergences.

4 Addressing ill-defined KL divergence

Let $Z_Q$ and $Z_P$ be two random variables with distribution $Q_Z$ and $P_Z$. The KL divergence between $Q_Z$ and $P_Z$ is not defined if $Q_Z$ or $P_Z$ does not have valid density function. Let $K$ be an a.c. random variable that is independent of $Z_Q$ and $Z_P$ and has density $p_K$. We define $Z_P = Z_P + K; Z_Q = Z_Q + K$ with distributions $P_Z$ and $Q_Z$ respectively. Then $P_Z$ and $Q_Z$ are a.c. with density functions

$$q(\tilde{z}) = \int p_K(\tilde{z} - z)dQ_Z \quad p(\tilde{z}) = \int p_K(\tilde{z} - z)dP_Z.$$

(6)

The spread KL divergence between $Q_Z$ and $P_Z$ as the KL divergence between $Q_Z$ and $P_Z$ is:

$$\text{KL}(Q_Z||P_Z) \equiv \text{KL}(Q_Z||\bar{P}_Z) \equiv \text{KL}(q(\tilde{z})||p(\tilde{z})).$$

(7)

In this work we let $K$ be a Gaussian with diagonal covariance $\sigma^2 Z I$ to satisfy the sufficient conditions such that $KL$ is a valid divergence (see [45] for details) and has the properties:

$$\text{KL}(Q_Z||P_Z) \geq 0, \quad \text{KL}(Q_Z||P_Z) = 0 \iff Q_Z = P_Z.$$

(8)

Since $Q_Z$ and $P_Z$ are transformed from $P_d$ and $P_X$ using an invertible function $g$, we have

$$Q_Z = P_Z \iff P_d = P_X.$$

(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.

5 Identifiability of intrinsic dimension

A byproduct of our model is that the intrinsic dimension of the data manifold can be identified. Section 4 shows that when $\text{KL}(Q_Z||P_Z) = 0 \iff P_X = P_d$, the supports of $P_X$ and $P_d$ will also have the same intrinsic dimension: $\text{Indim}(P_X) = \text{Indim}(P_d)$. The flow function $g$, and its inverse $g = f^{-1}$, are bijective and continuous so $f$ is a diffeomorphism [22]. Due to the invariance of dimension property of diffeomorphisms [28, Theorem 2.17], the manifold that supports $P_Z$ will have the same dimension as the manifold that supports $P_X$ thus

$$\text{Indim}(P_Z) = \text{Indim}(P_X) = \text{Indim}(P_d).$$

(10)

Since the intrinsic dimension of $P_Z$ is equal to the rank of the matrix $AA^T$, we can calculate $\text{Rank}(AA^T)$ 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

$$\text{Indim}(P_d) = \text{Rank}(AA^T).$$

(11)

We have shown that we can identify the intrinsic dimension of $P_d$ using the spread KL divergence. In the following section, we discuss how to estimate this spread in practice.

6 Estimation of the spread KL divergence

Our goal is to minimize the spread KL divergence between $Q_Z$ and $P_Z$. Using our definition of spread divergence (equation 7), this is equivalent to minimizing

$$\text{KL}(q(\tilde{z})||p(\tilde{z})) = \int q(\tilde{z}) \log q(\tilde{z})d\tilde{z} - \int q(\tilde{z}) \log p(\tilde{z})d\tilde{z}.$$

(12)

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

**Term 1**: We use $H(\cdot)$ to denote the differential entropy. Term 1 is the negative entropy $-H(Z_Q)$. For a volume preserving $g$ and $X_d$ is a.c., the entropy $H(Z_Q) = H(X_d)$ and is independent of the
model parameters and can be ignored during training. However, the entropy \( H(Z_Q) = H(Z_Q + K) \) 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 the \( H(Z_Q) \) and volume preserving function \( g \) is weak, thus we can approximate equation 12 by leaving out term 1 and will not affect the training.

To build intuitions, we first assume \( X_d \) is a.c., so \( Z_Q = g(X_d) \) is also a.c.. Using standard entropy properties \([23]\), we can pose the following relationship

\[
H(Z_Q) \leq H(Z_Q + K) = H(Z_Q) + I(Z_Q + K, K),
\]

where \( I(\cdot, \cdot) \) denotes the mutual information. Since \( Z_Q \) is independent of function \( g \) and if \( \sigma^2_Z \to 0 \), then \( I(Z_Q + K, K) \to 0 \) (see Appendix B.2 for a proof), the contribution of the \( I(Z_Q + K, K) \) term, with respect to training \( g \), becomes negligible in the case of small \( \sigma^2_Z \).

Unfortunately, equation 13 is no longer valid when \( P_d \) lies on a manifold since \( Z_Q \) will be a singular random variable and the differential entropy \( H(Z_Q) \) is not defined. In Appendix B.3, we show that leaving out the entropy \( H(Z_Q) \) corresponds to minimizing an upper bound of the spread KL divergence. To further find out the contribution of the negative entropy term, we compare between leaving out \(-H(Z_Q) \) and approximating \(-H(Z_Q) \) during training. In Appendix B.4, we discuss the approximation technique and give the empirical evidence which shows that the ignoring \(-H(Z_Q) \) will not affect the training of our model. Therefore, we make use of volume preserving \( g \) and small variance \( \sigma^2_Z = 1 \times 10^{-4} \) in our experiments.

In contrast to other volume preserving flows, that utilize a fixed prior \( P_Z \), 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 or abilities 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 D.1).

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

\[
p(\tilde{z}) = \mathcal{N}(0, AA^T + \sigma^2_Z I).
\]

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

\[
\int q(\tilde{z}) \log p(\tilde{z})d\tilde{z} \approx \frac{1}{N} \sum_{n=1}^{N} \log p(\tilde{z}_n),
\]

where \( q(\tilde{z}) = \int p(\tilde{z}|z)dQ_Z \). To sample from \( q(\tilde{z}) \), we first get a data sample \( x \sim P_d \), use function \( g \) to get \( z = g(x) \) (so \( z \) is a sample of \( Q_Z \)) and finally sample \( \tilde{z} \sim p(\tilde{z}|z) \).

7 Comparisons with related work

The study of manifold learning for nonlinear dimensionality reduction \([7]\) and intrinsic dimension estimation \([6]\) is a rich field with an extensive set of tools. However, most methods do not model data density on the manifold and are thus not applicable for the same purposes as the models introduced in our work. Popular density-based probabilistic models such as Variational Auto-Encoder \([21]\) or Flow variants \([20, 35, 11, 26, 38]\) assume the data distribution is a.c. and as such cannot model distributions that lie on a low-dimensional manifold. A number of recent works do however introduce normalizing flows on manifolds such as Riemannian Flows \([14, 36]\) and M-Flow \([5]\). Table 1 provides a succinct comparison of properties, for the outlined approaches, in contrast with our current work. Appendix C provides further discussion on these differences. We believe our approach is the first method capable of modeling (and identifying) distributions that lie on a (continuous) low-dimensional manifold, without knowledge of related charts or intrinsic manifold dimension.

8 Experiments

Traditional flow models assume that a data distribution is absolutely continuous and we extend this assumption such that the data distribution lies on one continuous manifold. We demonstrate
Table 1: Flow-based comparisons across two model properties: (1) is it learnable that the distribution lies on a low-dimensional manifold (which is not a.c.) and (2) model requirements for learning the manifold distribution. See Appendix C for a detailed discussion.

| Flow Models | Manifold | Requirements | Intrinsic Dimension | Chart |
|-------------|----------|--------------|---------------------|-------|
| RealNVP [11] / GLOW [20] / GIN [38] | ✗ | - | - | - |
| Relaxed Injective Flow [26] | ✗ | - | - | - |
| Riemannian Flow [14, 36] | ✓ | Required | Required | Required |
| M-flow [5] | ✓ | Required | Not required | |
| Ours | ✓ | Not required | Not required | |

Figure 3: (a) shows the samples from the data distribution $P_d$ (blue) and model $P_X$ (red). (b) shows the sample from the learned prior $P_Z$ (blue) and the distribution $Q_Z$ (red). (c) shows the eigenvalues of $AA^T$. (d) shows true density on the manifold. (e) shows density estimation on the manifold.

both the effectiveness and robustness of our model by considering firstly (1) data distributions that satisfy our extended assumption: toy 2D and 3D data, the fading square dataset; and secondly (2) distributions where our assumption no longer holds: synthesized and real MNIST [27], SVHN [32], CIFAR10 [24] and CelebA [29]. Our flow networks utilize incompressible affine coupling layers, introduced by [38, 11], for toy and MNIST experiments. For the color image experiments, we alternatively use a volume preserving variation of the Glow structure [20]. Further experimental details are found in Appendix D.

8.1 Synthetic data

2D toy data We firstly verify our method using the toy dataset described in Section 2 and Figure 1a. The flow function has two coupling layers. Figure 3 shows model samples, the learned prior and the eigenvalues of $AA^T$. We observe the sample quality improves upon those in Figure 1b and that the prior has learned a degenerate Gaussian with $\text{Indim}(P_Z) = 1$, matching $\text{Indim}(P_d)$. 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 D.2 for further details.

3D toy data (S-curve) We next consider the S-curve dataset shown in Figure 4a. The data distribution lies on a 2D manifold in a 3D space, therefore $\text{Indim}(P_d) = 2$. See Appendix D.1 for training details. After training, our model learns a function $g$ to transform $P_d$ to $Q_Z$, where the latter lies on a 2D linear subspace in 3D space (see Figure 4b). Next, a linear dimensionality reduction can be conducted to generate 2D data representations. We briefly outline the procedure here. For $Q_Z$ with $\text{Amdim}(Q_Z) = D$ and $\text{Indim}(Q_Z) = K$, we firstly find the eigenvectors $e_1, \ldots, e_D$ of $AA^T$ as sorted by their eigenvalues. When $\text{Rank}(AA^T) = K \leq D$, there exist $K$ eigenvectors with positive eigenvalues. We select the first $K$ eigenvectors and form the matrix $E = [e_1, \ldots, e^K]$ with dimension $D \times K$. We then transform each data sample $x \in \mathbb{R}^D$ into $Z$ space: $z = g(x)$, such that $z \in \mathbb{R}^K$. Finally, a linear projection is carried out $z^{\text{proj}} = zE$ to obtain the lower dimensional representation $z^{\text{proj}} \in \mathbb{R}^K$. This procedure can be seen as a nonlinear PCA, where the nonlinear component is solved using the learned function $g$.

Figure 4b contains resulting representations. 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 pre-
Figure 4: (a) S-curve data samples $x \sim P_d$. (b) The latent representation $z = g(x)$, points can be observed to lie on a linear subspace. (c) Eigenvalues of the matrix $AA^T$, we deduce that $\text{Indim}(P_d) = 2$. (d) Our representation after the dimensionality reduction $z^{\text{proj}} = zE$.

Figure 5: (a) and (b) show samples from the data distribution and our model, respectively. (c) shows that a traditional flow based model with fixed Gaussian prior fails to generate any valid samples. (d) the first 20 eigenvalues of the matrix $AA^T$. (e) visualization of the representation after applying dimensionality reduction. See text for further discussion.

serves the structure of the original distribution. In contrast, we find that a flow with fixed Gaussian prior fails to learn such a data distribution, see Figure 4b.

Fading Square dataset The fading square dataset [37] was proposed in order to assess model behavior when data distribution and model possess differing intrinsic dimension and therefore affords a further relevant test of our work. The dataset consists of $32 \times 32$ images with $6 \times 6$ grey squares on a black background. The grey scale values are sampled from a uniform distribution with range $[0, 1]$, so $\text{Indim}(P_d)=1$. Figure 5a shows samples from the dataset to which we fit our model and additional model details may be found in Appendix D. 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 that we have successfully identified the intrinsic dimension of $P_d$. We further carry out the dimensionality reduction process; the latent representation $z$ is projected onto a 1D line and we visualize the correspondence between the projected representation and the data in Figure 5e. Pixel grey-scale values can be observed to decay as the 1D representation is traversed from left to right, indicating that our representations are consistent with the properties of the original data distribution. In contrast, we find that the traditional flow model, with fixed 1024D Gaussian $p(z)$, fails to learn the data distribution; see Figure 5c.

Synthetic MNIST We further investigate model training using images of digits. It may be noted that, for real-world image datasets, the true intrinsic dimension is unknown. Therefore in order to verify the correctness of our model’s ability to identify intrinsic data dimension, we firstly construct a synthetic dataset by fitting an implicit model $p_\theta(x) = \int \delta(x - g(z)p(z))dz$ to the MNIST dataset, and then sample from the trained model $x \sim p_\theta(x)$ in order to generate synthetic training data. The intrinsic dimension of the training dataset can then be approximated\(^4\) via the dimension of the

\(^4\)For implicit model: $X = g(Z)$, the intrinsic dimension of the model distribution will be not-greater-than the dimension of the latent variable: $\text{Indim}(X) \leq \dim(Z)$ [2]. Further, if strictly $\text{Indim}(X) < \dim(Z)$ this results in a ‘degenerated’ case. When we fit the model on a data distribution where $\text{Indim}(X) \geq \dim(Z)$, we
latent random variable $Z$: $\text{Indim}(P_d) = \text{dim}(Z)$. We construct two datasets with $\text{dim}(Z)=5$ and $\text{dim}(Z)=10$ such that $\text{Indim}(P_d)=5$ and $\text{Indim}(P_d)=10$, 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 that when the continuous manifold assumption is not satisfied, training instabilities can occur. Towards alleviating this issue, we smooth the data manifold by adding small Gaussian noise to the training data (standard deviation $\sigma_x=0.05$, in our experiments). We note that adding Gaussian noise changes the intrinsic dimension of the training distribution towards alignment with the ambient dimension and therefore mitigate this undesirable effect by annealing $\sigma_x$ after $2000k$ iterations with a factor of 0.9 every $10k$ iterations. 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 datasets with intrinsic dimension 5 and 10. It can be observed that 5 and 10 eigenvalues are significantly larger than the remaining values, respectively and we can thus conclude that the intrinsic dimension of the two datasets are 5 and 10. Remaining non-zero eigenvalues can be attributed to the small Gaussian noise added to the training data. Therefore, our method can successful model complex data distribution and identify the true intrinsic dimension when the data distribution lies on a continuous manifold. In the next section, we apply our model to real image data where this assumption may not be satisfied.

8.2 Real world images

Real MNIST For the original MNIST dataset, it was previously shown that digits have differing intrinsic dimension [10]. This suggests the MNIST distribution may lie on several, disconnected manifolds with different intrinsic dimension. Although our model assumes that $P_d$ 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 data and plot the eigenvalues in Figure 6f. In contrast to Figures 6d and 6e, the gap between eigenvalues predictably exhibits a less obvious step change. However the values suggest that the intrinsic dimension of MNIST is between 11 and 14. This result is consistent with previous estimations stating that the intrinsic dimension of MNIST is between 12 and 14 [13, 17, 10]. Recent work [9] discusses fitting flow models to a $P_d$ 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, which we consider a promising direction for future work. Figure 6 also provides model samples. We compared samples with a non-volume preserving flow with a fixed prior and found our model can provide improved sample quality, see Appendix D.5.

Color images We conduct further experiments on real-world color images: SVHN, CIFAR10 and CelebA. Data samples from SVHN and CIFAR10 are represented by $32 \times 32 \times 3$ dimension vectors and for CelebA our pre-processing involves centre-cropping each image and resizing to $32 \times 32 \times 3$. Pixels take discrete values from $[0, \ldots, 255]$ and we follow a standard dequantization process [40]: $x = \frac{x+\text{Uniform}(0,1)}{256}$ to transform each pixel to a continuous value in $[0,1]$. The manifold structure of these image datasets has been lesser 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 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 D.6. Figure 7 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 [43]. 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

---

*assume that degeneration will not occur during training, resulting in $\text{Indim}(P_d) \approx \text{dim}(Z)$. For simplicity, we equate $\text{Indim}(P_d) = \text{dim}(Z)$ in the main text.*
training trends: a latent variable model, e.g., VAE usually requires very large latent dimension to generate high-quality images \[41, 8\]. For example, in the state-of-the-art image generation work \[41\], a VAE with latent size 153600 is used to fit a CIFAR10 dataset with shape \(32 \times 32 \times 3\). These considerations lead us to believe that the estimation of intrinsic dimension for natural images is 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 downstream latent variable models. We leave deeper study of the structure and geometry of image manifolds to future work.

\[\text{Figure 7: Samples and eigenvalues for three models. We can find approximately half of the eigenvalues converges to 0 but there is a large number of eigenvalues (≈ 1500) greater than 0.}\]
9 Conclusion

In this work we present a principled strategy towards enabling flow based models to learn data distributions that lie on a continuous low-dimensional manifold and identify the intrinsic dimension of the data manifold at the same time. Our findings show the benefits of our approach in terms of sample generation and representation quality and we also expose new potential directions towards more rigorously investigating the cases when data distributions fail to satisfy common flow assumptions.

References

[1] S. M. Ali and S. 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.

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

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

[4] D. Beymer and T. Poggio. Image representations for visual learning. *Science*, 272(5270):1905–1909, 1996.

[5] J. Brehmer and K. Cranmer. Flows for simultaneous manifold learning and density estimation. *arXiv preprint arXiv:2003.13913*, 2020.

[6] F. Camasta and A. Staiano. Intrinsic dimension estimation: Advances and open problems. *Information Sciences*, 328:26–41, 2016.

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

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

[9] R. Cornish, A. L. Caterini, G. Deligiannidis, and A. Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. *arXiv preprint arXiv:1909.13833*, 2019.

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

[11] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[12] R. Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

[13] E. Facco, M. d’Errico, A. Rodriguez, and A. Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):1–8, 2017.

[14] M. C. Gemici, D. Rezende, and S. Mohamed. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*, 2016.

[15] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[16] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[17] M. Hein and J.-Y. Audibert. Intrinsic dimensionality estimation of submanifolds in rd. In *Proceedings of the 22nd international conference on Machine learning*, pages 289–296, 2005.

[18] M. F. Huber, T. Bailey, H. Durrant-Whyte, and U. 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.
[19] J.-H. Jacobsen, A. Smeulders, and E. Oyallon. i-revnet: Deep invertible networks. arXiv preprint arXiv:1802.07088, 2018.

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

[21] D. P. Kingma and M. Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[22] I. Kobyzev, S. Prince, and M. Brubaker. Normalizing flows: An introduction and review of current methods. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020.

[23] I. Kontoyiannis and M. Madiman. Sumset and inverse sumset inequalities for differential entropy and mutual information. IEEE transactions on information theory, 60(8):4503–4514, 2014.

[24] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[25] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[26] A. Kumar, B. Poole, and K. Murphy. Regularized autoencoders via relaxed injective probability flow. In International Conference on Artificial Intelligence and Statistics, pages 4292–4301. PMLR, 2020.

[27] Y. LeCun. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998.

[28] J. M. Lee. Smooth manifolds. In Introduction to Smooth Manifolds, pages 1–31. Springer, 2013.

[29] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), December 2015.

[30] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), December 2015.

[31] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[32] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[33] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. arXiv preprint arXiv:1912.02762, 2019.

[34] A. Papoulis and S. U. Pillai. Probability, random variables, and stochastic processes. Tata McGraw-Hill Education, 2002.

[35] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. arXiv preprint arXiv:1505.05770, 2015.

[36] D. J. Rezende, G. Papamakarios, S. Racanière, M. S. Albergo, G. Kanwar, P. E. Shanahan, and K. Cranmer. Normalizing flows on tori and spheres. arXiv preprint arXiv:2002.02428, 2020.

[37] P. K. Rubenstein, B. Schoelkopf, and I. Tolstikhin. On the latent space of wasserstein auto-encoders. arXiv preprint arXiv:1802.03761, 2018.

[38] P. Sorrenson, C. Rother, and U. Kothe. Disentanglement by nonlinear ica with general incompressible-flow networks (gin). arXiv preprint arXiv:2001.04872, 2020.

[39] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf. Wasserstein auto-encoders. arXiv preprint arXiv:1711.01558, 2017.

[40] B. Uria, I. Murray, and H. Larochelle. Rnade: The real-valued neural autoregressive density-estimator. arXiv preprint arXiv:1308.0186, 2013.
[41] A. Vahdat and J. Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

[42] R. W. Yeung. *Information theory and network coding*. Springer Science & Business Media, 2008.

[43] K. Yu and T. Zhang. Improved local coordinate coding using local tangents. In *ICML*. Citeseer, 2010.

[44] M. Zhang, T. Bird, R. Habib, T. Xu, and D. Barber. Variational f-divergence minimization. *arXiv preprint arXiv:1907.11891*, 2019.

[45] M. Zhang, P. Hayes, T. Bird, R. Habib, and D. Barber. Spread Divergences. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, 2020.

[46] S. Zhao, J. Song, and S. Ermon. Towards deeper understanding of variational autoencoding models. *arXiv preprint arXiv:1702.08658*, 2017.
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 $p(x)$ to the data. A popular choice to achieve this involves minimization of the KL divergence where:

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

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

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

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

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 $f: Z \rightarrow X$, $g = f^{-1}$, 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 data distribution and define $Z_q : Z_q = g(X_d)$ with density $q(z)$, so $q(z)$ can be represented as

$$q(z) = \int \delta(z - g(x)) p_d(x) dx.$$  \hspace{1cm} (19)

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

$$\text{KL}(q(z)||p(z)) = \int q(z) \log q(z) dz - \int q(z) \log p(z) dz.$$ \hspace{1cm} (20)

Term 1: using the properties of transformation of random variable [34, pp. 660], the negative entropy can be written as

$$\int q(z) \log q(z) dz = \int p_d(x) \log p_d(x) dx - \int p_d(x) \log \left| \det \left( \frac{\partial g}{\partial x} \right) \right| dx.$$ \hspace{1cm} (21)

Term 2: the cross entropy can be written as

$$\int q(z) \log p(z) dz = \int \int \delta(z - g(x)) p_d(x) \log p(z) dz dx$$ \hspace{1cm} (22)

$$= \int p_d(x) \log p(g(x)) dx.$$ \hspace{1cm} (23)

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

$$\text{KL}(q(z)||p(z)) = \text{KL}(p_d(x)||p(x)).$$ \hspace{1cm} (24)

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 $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $g$ is a volume preserving flow with parameter $\theta$. Let $Z_1 = g_{\theta_1}(Z)$ be a Gaussian with covariance $\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$ and $Z_2 = g_{\theta_2}(X)$ be

---

6For simplicity, we use notation $p(x)$ to represent the model $p_X(x)$ unless otherwise specified.
a Gaussian with covariance $\begin{bmatrix} 3 & 0 \\ 0 & 1/3 \end{bmatrix}$. Therefore the entropy $H(Z_1) = H(Z_2) = H(X)$ and doesn’t not depend on $\theta$. Let $K$ be an Gaussian with zero mean and covariance $\begin{bmatrix} 1 & 0 \\ 0 & 1/2 \end{bmatrix}$, so $Z_1 + K$ is a Gaussian with covariance $\begin{bmatrix} 3 & 0 \\ 0 & 3/2 \end{bmatrix}$ and $Z_2 + K$ is a Gaussian with covariance $\begin{bmatrix} 4 & 0 \\ 0 & 4/3 \end{bmatrix}$. Therefore $H(Z_1 + K) \neq H(Z_2 + K)$ and $H(g_\theta(X) + K)$ depends on $\theta$. 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

$$I(Z + K, K) = H(Z + K) - H(Z + K|K)$$

(25)

$$= H(Z + K) - H(Z|K)$$

(26)

$$= H(Z + K) - H(Z).$$

(27)

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

$$H(Z) \leq H(Z + K) = H(Z) + I(Z + K, K).$$

(28)

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

$$I(Z + K, K) = H(Z + K) - H(Z) = 0,$$

(29)

this is because the mutual information between an a.c. random variable and a singular random variable is still well defined, see [42, 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 $\sigma_1^2 = \sigma_2^2 + \sigma_3^2$, and let $K_\delta$ be the random variable of a Gaussian that has 0 mean and variance $\sigma_3^2$ such that $K_1 = K_2 + K_\delta$. By the data-processing inequality, we have

$$I(Z + K_2, K_2) \leq I(Z + K_2 + K_\delta, K_2 + K_\delta) = I(Z + K_1, K_1).$$

(30)

Therefore, $I(Z + K, K)$ 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 $H(Z_\tilde{Q})$ in equation 12 is equivalent to minimizing an upper bound of the spread KL divergence.

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

$$H(Z_\tilde{Q} + K) - H(K) = H(Z_\tilde{Q} + K) - H(Z_\tilde{Q} + K|Z_\tilde{Q})$$

(31)

$$= I(Z_\tilde{Q} + K, Z_\tilde{Q}) \geq 0.$$ 

(32)

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 [42, Theorem 10.33] for a proof.
Therefore, we can construct an upper bound of the spread KL objective in equation 12

\[
\text{KL}(\tilde{q}||\tilde{p}) = \int q(\tilde{z}) \log q(\tilde{z}) d\tilde{z} - \int q(\tilde{z}) \log p(\tilde{z}) d\tilde{z} - H(K)
\]

\[
\leq -H(K) - \int q(\tilde{z}) \log p(\tilde{z}) d\tilde{z}. \quad (33)
\]

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

\[
-I(ZQ + K, ZQ) \geq 0 \quad (34)
\]

\[
\leq -H(K) - \int q(\tilde{z}) \log p(\tilde{z}) d\tilde{z}. \quad (35)
\]

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 for the training. The negative entropy of random variable \(Z\) is

\[
-H(Z) = \int q(\tilde{z}) \log q(\tilde{z}) d\tilde{z}, \quad (36)
\]

where

\[
q(\tilde{z}) = \int p_K(\tilde{z} - z) dQ_Z, \quad (37)
\]

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

\[
q(\tilde{z}) \approx \frac{1}{N} \sum_{n=1}^{N} \mathcal{N}(\tilde{z}; z^n, \sigma^2_Z I) \equiv q^N(\tilde{z}) \quad (38)
\]

where \(z^n\) is the \(n\)th sample from distribution \(Q\) by first sampling \(x^n \sim P_d\) and letting \(z^n = g(x^n)\). We denote the random variable of this Gaussian mixture as \(Z^N\) and approximate

\[
-H(Z) \approx -H(Z^N), \quad (39)
\]

However, the entropy of a Gaussian mixture distribution does not have a closed form, so we further conduct a first order Taylor expansion approximation [18]

\[
-H(Z^N) \approx \frac{1}{N} \sum_{n=1}^{N} \log q^N(\tilde{z} = z^n), \quad (40)
\]

this approximation is accurate for small \(\sigma^2_Z\). Finally we have our approximation

\[
-H(Z) \approx \frac{1}{N} \sum_{n=1}^{N} \log q^N(\tilde{z} = z^n). \quad (41)
\]

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 tracking of the value of the entropy \(H(Z)\) (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 D.
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 tracking of the value of the entropy $H(Z)$ (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 D.

In Figure 8, we plot the (approximated) entropy value $H(Z)$ 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 theoretical investigation on the effects of leaving out the entropy term during training to future work.

In Figure 8, we plot the (approximated) entropy value $H(Z)$ 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 theoretical investigation on the effects of leaving out the entropy term during training to future work.

### C Related Work

Classic latent variable generative models assume that data distributions lie around a low-dimensional manifold, for example the Variational Auto-Encoder [21] and the recent introduced Relaxed Injective Flow [26] (it uses an L2 reconstruction loss in their objective, which implicit assume the model has a Gaussian observation distribution with isotropic variance.). Such methods typically assume that observational noise is not degenerated (e.g. a fixed Gaussian distribution), therefore the model distribution is absolutely continuous and maximum likelihood learning is thus well defined. However, common distributions such as natural images usually don’t have Gaussian observation noise [46]. Therefore, we focus on modeling distributions that lie on a low-dimensional manifold.

The study of manifold learning for nonlinear dimensionality reduction [7] and intrinsic dimension estimation [6] is a rich field with an extensive set of tools. However, most methods commonly do not model data density on the manifold and are thus not used for the same purpose as the models introduced here. There are however a number of recent works that introduced normalizing flows on manifolds that we now highlight and relate to our approach.

Several works define flows on manifolds with prescribed charts. [14] generalized flows from Euclidean spaces to Riemannian manifolds by proposing to map points from the manifold $M$ to $\mathbb{R}^K$, apply a normalizing flow in this space and then map back to $M$. The technique has been further extended to Tori and Spheres [36]. These methods require knowledge of the intrinsic dimension $K$ and a parameterization of the coordinate chart of the data manifold.

Without providing a chart mapping a priori, M-flow [5] recently proposed an algorithm that learns the chart mapping and distribution density simultaneously. However, their method still requires the dimensionality of the manifold is known. They propose that the 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 wild behavior of the optimization process since these terms are not well defined.

While the GIN model considered in [38] could recover the low dimensional generating latent variables following their identifiability theorem, the assumptions therein require knowledge of an auxiliary variable, e.g. the label, which is not required in our model. Behind this superficial 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.

D Experiments

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

D.1 Network architecture for toy and MNIST dataset

The flow network we use consists of incompressible affine coupling layers [38, 11]. Each coupling layer splits a $D$-dimensional input $x$ into two parts $x_{1:d}$ and $x_{d+1:D}$. The output of the coupling layer is

$$y_{1:d} = x_{1:d}$$

$$y_{d+1:D} = x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d}),$$

where $s : \mathbb{R}^d \rightarrow \mathbb{R}^{D-d}$ and $t : \mathbb{R}^d \rightarrow \mathbb{R}^{D-d}$ 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 $\sum_j s(x_{1:d})_j$. To make the coupling transform volume preserving, we normalize the output of the scale function, so the $i$-th dimension of the output is

$$\tilde{s}(x_{1:d})_i = s(x_{1:d})_i - \frac{1}{D-d} \sum_j s(x_{1:d})_j,$$

and the log-determinant of the Jacobian is $\sum_i \tilde{s}(x_{1:d})_i = 0$. We compare a volume preserving flow with a learnable prior (normalized $s(\cdot)$) to a non-volume preserving flow with fixed prior (un-normalized $s(\cdot)$). 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 $3 \times 3$ and padding size 1. The activation is Leaky ReLU with 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 [38, 19]. When multiple convolutional nets are connected together, we randomly permute the channels of the output of each network except the final one. In Table 2, 3, 4, 5, we show the network structures for our four main paper experiments.

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

Table 2: The network structure for toy datasets.
Table 3: The network structure for S-Curve dataset. If we denote the input vector of each coupling is \( x \), the function \( s \) and \( t \) takes \( x[0] \) in the first coupling layer and \( x[1] \) in the second coupling layer.

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

Table 4: The network structure for our fading square dataset experiments. If we denote the input vector of each coupling to be \( x \), the functions \( s \) and \( t \) take values \( x[1:2] \) in the even coupling layer and \( x[3] \) in the odd coupling layer.

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

Table 5: Network structure for synthetic MNIST dataset experiment.

D.2 Toy dataset

We also construct a second dataset and train a flow model using the same training procedure discussed in Section 8. Figure 9a shows the samples from the data distribution \( \mathbb{P}_d \), each data point is a 2D vector \( x = [x_1, x_2] \) where \( x_1 \sim N(0, 1) \) and \( x_2 = x_1 \), so \( \text{Indim}(\mathbb{P}_d) = 1 \). Figure 9f shows that the prior \( \mathbb{P}_Z \) has learned the true intrinsic dimension \( \text{Indim}(\mathbb{P}_Z) = \text{Indim}(\mathbb{P}_d) = 1 \). 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 9. 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 9c.

We also plot the ‘density allocation’ on the manifold for the two toy datasets. For example, for the data generation process \( x = [x_1, x_2] \) where \( x_1 \sim N(0, 1) \) 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 \( x_s \), uniformly from the support of the data distribution, the subscript ‘s’ here means that they only contain the information of the support. Specifically, since \( x_s = [x_1^s, x_2^s] \), we sample \( x_1^s \sim p = \mathcal{U}(-3\sigma, 3\sigma) \) (\( \sigma \) is the standard deviation of \( N(0, 1) \), \( \mathcal{U} \) 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 8 to obtain the projected samples \( z^{proj} \), so \( z^{proj} \in \mathbb{R}^{\text{Indim}(\mathbb{P}_d)} \). We also project the learned prior \( \mathbb{P}_Z \) to \( \mathbb{R}^{\text{Indim}(\mathbb{P}_d)} \) by constructing \( z^{proj} \) as a Gaussian with zero mean and a diagonal covariance contains the non-zeros eigenvalues of \( AA^T \). Therefore \( z^{proj} \) is a.s. in \( \mathbb{R}^{\text{Indim}(\mathbb{P}_d)} \) and we denote its density function as \( p^{proj}(z) \). We then use the density value \( p^{proj}(z = z^{proj}) \) to indicate the ‘density allocation’ at the location of \( x_s \) on the manifold support. In Figure 10, we compare our model with the ground truth and find that we can successfully capture the ‘density allocation’ on the manifold.
Figure 9: (a) shows the samples from the data distribution. (b) and (d) show samples from a flow with a fixed Gaussian prior and our method, respectively. (c) and (d) show the latent space in both models. In (f), we plot the eigenvalues of the matrix $AA^T$.

D.3 S-Curve dataset

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

We compare our method with a traditional normalizing flow with a fixed 3D Gaussian prior. Both models have the same network architecture and training procedure. Figure 11a and 11b show the samples from our model and the traditional flow with a fixed 3D Gaussian prior. We can observe more samples lying outside the true data distribution in Figure 11b than in Figure 11a. We can conclude that our model has better sample quality considering this S-Curve dataset. We also compare the latent representation for both models, see Figure 11c and 11d. We can see the representation distribution $Q_Z$ captures the structure of the data distribution well whereas the distribution $Q_Z$ in Figure 11d is unable to do so.

D.4 Fading Square dataset

To fit the data, we train our model for $20k$ iterations with batch-size 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 $1k$ iterations. We additionally use an $L2$ weight decay with factor 0.1.

D.5 MNIST Dataset

D.5.1 Implicit data generation model

To fit an implicit model to the MNIST dataset, we first train a Variational Auto-Encoder (VAE) ([21]) with Gaussian prior $p(z) = \mathcal{N}(0, I)$. The encoder is $q(z|x) = \mathcal{N}(\mu_\theta(x), \Sigma_\theta(x))$ where $\Sigma$ is a diagonal matrix. Both $\mu_\theta$ and $\Sigma_\theta$ are parameterized by a 3-layer feed-forward neural network with
Figure 10: (a) and (c) shows the ground truth ‘density allocation’ on the manifold for two toy datasets, (b) and (d) shows the ‘density allocation’ learned by our models.

A ReLU activation and the size of the two hidden outputs are 400 and 200. We use a Gaussian decoder $p(x|z) = \mathcal{N}(g_\theta(z), \sigma^2_x I)$ with fixed variance $\sigma_x = 0.3$. 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

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

see [21] for further details. We use an Adam optimizer with 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 $z \sim p(z)$ and letting $x = g_\theta(z)$. This is equivalent to taking a sample from an implicit model $p_\theta(x) = \int \delta(x - g_\theta(z)) d(z) dz$, see [39] for further discussion regarding this implicit model construction. In Figure 12, we plot samples from the trained implicit model with $\dim(z) = 5$, $\dim(z) = 10$ and the original MNIST data.

D.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 300$k$ iterations and, following the initial 100$k$ iterations, the learning rate decays every 10$k$ iterations by a factor 0.9.

In Figure 13, we plot the samples from our models trained on three different training datasets. In Figure 14, we also plot the samples from traditional flow models that trained on these three datasets using the same experiment settings, we found the samples from our models are sharper than that from traditional flow models.
Figure 11: Figure (a) and (b) plot the data distribution $\mathbb{P}_Z$ of the S-Curve dataset and the samples from our model and a traditional flow with a fixed Gaussian prior. Figure (c) shows the representation distribution $Q_Z$ and the learned prior $\mathbb{P}_Z$ of our model. In Figure (d), we plot the representation distribution $Q_Z$ using the flow with a fixed Gaussian prior.

Figure 12: Training data for the flow model. Figure (a) and (b) are synthetic MNIST samples from two implicit models with latent dimension 5 and 10. Figure (c) are samples from the original MNIST dataset.
Figure 13: Samples from our methods. Figure (a) and (b) are samples flow models trained on synthetic MNIST data with intrinsic dimension 5 and 10. Figure (c) are samples from a flow model that trained on original MNIST data.

Figure 14: Samples from traditional non-volume preserving flow models with fixed Gaussian prior.

D.6 Color image experiments

Our network for color image datasets is based on an open source implementation\textsuperscript{10}, where we modify the Glow [20] structure to make it volume preserving. The volume preserving affine coupling layer is similar to the incompressible flow, discussed in Section ?? of our main manuscript, however here the $s(\cdot)$ function takes the form:

$$s(\cdot) = \text{scale} \ast \tanh(\text{NN}(\cdot))$$

(45)

where ‘scale’ are channel-wise learnable parameters and $\text{NN}(\cdot)$ is the neural network. For the act-norm function $y_{ij} = s \odot x_{ij} + b$, where the log-determinant for position $x_{ij}$ is $\text{sum}(\log|s|)$, we thus normalize the $\text{sum}(\log|s|) = \log|s| - \text{mean}(\log|s|)$, so $\text{sum}(\log|s|) = 0$. We use LU decomposition for the $1 \times 1$ convolution, where $W = PL(U + \text{diag}(s))$ with log-determinant $\log|\det(W)| = \text{sum}(\log|s|)$. We can then normalize $\log|s| = \log|s| - \text{mean}(\log|s|)$ 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.

\textsuperscript{10}https://github.com/chrischute/glow
Figure 15: Samples for three models.
E License

The CelebA [30] and SVHN [31] dataset is available for non-commercial research purposes only. We didn’t find licenses for CIFAR [25] and MNIST [27]. The CelebA dataset may contain personal identifications.

The code https://github.com/chrischute/glow we based on is under MIT License.

F Societal Impacts

In this paper, researchers introduce a novel generative modeling tool that affords benefits to both sample generation and representation quality, improving the expressive power of normalizing flows. Flow based image models, in particular, are currently gaining popularity and may soon find application in many down-stream real-world tasks involving e.g. visual data. Typical use cases include classification, detection and additional tasks involving extraction of semantically meaningful information from imagery or video. With this in mind our caution must remain in the fore, when considering related technology, to avoid it being harnessed to enable harmful ends. Identification or classification of people without their knowledge, towards control or criminalization provides an obvious example.