# Learning Geometry and Topology via Multi-Chart Flows

**Hanlin Yu**
University of Helsinki

**Søren Hauberg**
Technical University of Denmark

**Marcelo Hartmann**
University of Helsinki

**Arto Klami**
University of Helsinki

**Georgios Arvanitidis**
Technical University of Denmark

## Abstract

Real world data often lie on low-dimensional Riemannian manifolds embedded in high-dimensional spaces. This motivates learning degenerate normalizing flows that map between the ambient space and a low-dimensional latent space. However, if the manifold has a non-trivial topology, it can never be correctly learned using a single flow. Instead multiple flows must be 'glued together'. In this paper, we first propose the general training scheme for learning such a collection of flows, and secondly we develop the first numerical algorithms for computing geodesics on such manifolds. Empirically, we demonstrate that this leads to highly significant improvements in topology estimation.

## 1 INTRODUCTION

Normalizing flows (Papamakarios et al., 2021) are a family of flexible neural network models of complex probability distributions from finite observations. They are bijections that allow tractable sampling and log-density evaluations. Due to the bijectivity requirement, the dimensionalities of the latent and ambient spaces need to be identical. As real-world datasets often lie on a low-dimensional Riemannian manifold embedded in a high-dimensional ambient space, we usually do not expect a bijection. As such, the modeling strategy of the flows can be ill-posed. When the manifold structure is known beforehand, one can construct specific flows that adapt to it (Rezende et al., 2020; Ng and Zammit-Mangion, 2023).

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. 

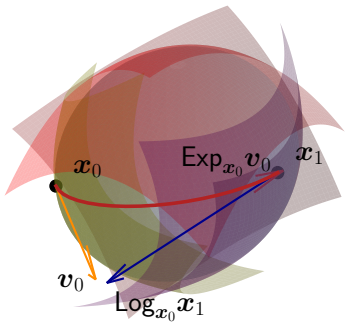


Figure 1: Using multiple charts to cover a manifold gives the ability to learn the manifold's true geometry and topology. The charts may not perfectly align, requesting careful treatments.

A probability density on an unknown Riemannian manifold can be modeled using a degenerate normalizing flow (Brehmer and Cranmer, 2020; Caterini et al., 2021), by learning a mapping between a low-dimensional latent space and a high-dimensional ambient space. It was demonstrated that, for certain manifolds, the flows are able to model the distribution accurately. However, these methods often consider a single normalizing flow, which is justified only when the manifold is diffeomorphic to a Euclidean space. Some common manifolds, for instance the sphere and the torus, do not satisfy this assumption. It is by definition impossible for a single-chart flow to model these manifolds correctly due to the topological mismatch.

Understanding the underlying topology and geometry of the data manifold using only finite observations is an open problem in machine learning (Hauberg, 2019; Moor et al., 2020; Acosta et al., 2023; Ross et al., 2024; Diepeveen, 2024). Some attempts focus on capturing the Riemannian structure using the pullback metric induced by a single normalizing flow (Hoffman et al., 2019; Kruiff et al., 2024), while other works (Schonsheck et al., 2020; Kalatzis et al., 2022; Sidheekh et al.,

2022; Alberti et al., 2024) use multiple autoencoders or normalizing flows to cover the manifold. However, these approaches often lack a principled way of choosing the correct encoder (Loaiza-Ganem et al., 2024). Most importantly, these works did not study the geometric and topological implications of the manifold represented by the generative model.

**In this paper**, we aim at learning data manifolds with the correct geometry and topology. Classic differential geometry achieves this by gluing together local "patches" of the manifold, each of which is a diffeomorphism. Similarly, we learn a mixture of flows, each corresponding to a "patch". In classic differential geometry, patches are assumed to overlap perfectly in bordering regions, but such theoretical constructions are unrealistic in practice using flows (see Figure 1). We propose a formalism to handle such non-overlapping patches, and further develop algorithms for computing geodesics over manifolds defined through such mixture of flows. Our contributions are:

1. We demonstrate that using a single normalizing flow to model a distribution on a Riemannian manifold can lead to significant misunderstanding in terms of the underlying topology and geometry.

2. We provide general training strategies for multi-chart flows based on the maximum likelihood principle and demonstrate that they can perform better than single-chart flows of similar complexities. While both gluing together multiple degenerate normalizing flows (Kalatzis et al., 2022; Sidheekh et al., 2022) and training a mixture of flows using Expectation Maximization (EM) (Ng and Zammit-Mangion, 2023) have been explored before, to the best of our knowledge, our work is the first to formulate the Maximum Likelihood Estimation / EM algorithm for directly training such a mixture of degenerate flows which also provides a principled way to choose the flows through the responsibilities.

3. While previous works generally focus on modeling and density estimation and **did not** study how to compute the exponential maps and logarithmic maps, we develop geodesic algorithms on the learned manifold. We provide specialized tools to utilize the geometry learned by multi-chart flows that crucially respect the actual topology of the underlying manifold, taking into account that the individual flows may not strictly agree with each other, yielding more reliable representations. This is our main contribution.

# 2 PRELIMINARIES

## 2.1 A Brief Intro to Riemannian Geometry

Riemannian manifolds (Do Carmo, 1992; Lee, John M., 2018) are spaces that locally resemble Euclidean spaces. We consider a $d$-dimensional Riemannian manifold isometrically embedded in a $D$-dimensional Euclidean space, following the typographic mnemonic notation that the lower-case symbol denotes the smaller dimensionality. The well-known Nash embedding theorem (Lee, John M., 2018) guarantees that such an embedding exists for all Riemannian manifolds, though it is not an operational definition. A *local coordinate chart* is given by the pair $(U, \phi)$, where $U$ is an open set on the manifold and $\phi$ is a bijection between $U$ and an open set of $d$-dimensional Euclidean space. For simple manifolds, one might be able to construct a single chart that covers the entire manifold. However, for more complex ones, we need a collection of local charts that transition smoothly, known as an *atlas*.

A curve with zero tangential acceleration, intuitively a shortest path on the manifold, is known as a *geodesic*. For a point $\boldsymbol{x}$ on a $d$-dimensional Riemannian manifold, one can attach a $d$-dimensional Euclidean space known as the tangent space at $\boldsymbol{x}$, containing all vectors that are tangent to the surface at $\boldsymbol{x}$. Given the initial position $\boldsymbol{x}_0$ and the initial velocity $\boldsymbol{v}_0$, one can use the exponential map $\mathrm{Exp}_{\boldsymbol{x}_0}(\boldsymbol{v}_0)$ to follow along the geodesic for unit time and obtain the final position $\boldsymbol{x}_1$. Given the initial position $\boldsymbol{x}_0$ and the final position $\boldsymbol{x}_1$, one can use the logarithmic map $\mathrm{Log}_{\boldsymbol{x}_0}(\boldsymbol{x}_1)$ to obtain (one of) the initial velocity that satisfies the boundary condition. These geometric operations are also illustrated in Figure 1, where we show that these quantities are well-defined even if the manifold is covered by multiple charts.

## 2.2 Normalizing Flows

A normalizing flow (Papamakarios et al., 2021) pushes forward a tractable distribution $p(\boldsymbol{u})$ from a *latent space* $\boldsymbol{U}$ to a complex distribution in the *input space* $\boldsymbol{X}$ through the bijection $\boldsymbol{f}$, such that the resulting distribution $q_{\boldsymbol{\theta}}(\boldsymbol{x})$ in $\boldsymbol{X}$ can be tractably evaluated.

For density estimation tasks on Euclidean spaces where $\dim(\boldsymbol{X}) = \dim(\boldsymbol{U}) = D$, normalizing flows are typically trained by minimizing the forward KL divergence $\mathrm{KL}\left(p(\boldsymbol{x}) | q_{\boldsymbol{\theta}}(\boldsymbol{x})\right)$ (Papamakarios et al., 2021), where the density $q_{\boldsymbol{\theta}}(\boldsymbol{x})$ can be evaluated as $q_{\boldsymbol{\theta}}(\boldsymbol{x}) = p(\boldsymbol{u}) |\det\left(\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{u})\right)|^{-1}\Big|_{\boldsymbol{u}=\boldsymbol{f}^{-1}(\boldsymbol{x})}$, with $\boldsymbol{J}_{\boldsymbol{f}} \in \mathbb{R}^{D\times D}$ being the Jacobian matrix of the parameterized transformation $\boldsymbol{f} : \boldsymbol{U} \to \boldsymbol{X}$; the dependency of $\boldsymbol{f}$ on $\boldsymbol{\theta}$ is dropped to avoid cluttering notations. One can employ

specific architectures, such that the log-determinant of the Jacobian can be obtained efficiently. Two notable examples are RealNVP flows (Dinh et al., 2017) and Neural Spline flows (Durkan et al., 2019).

In many scenarios, one would need to model distributions that are supported on Riemannian manifolds. In the following discussions, we largely follow Brehmer and Cranmer (2020) and Caterini et al. (2021). A normalizing flow on a Riemannian manifold is commonly parameterized as the composition of a series of $d$-dimensional layers $\boldsymbol{g}$ and a series of a $D$-dimensional layers $\boldsymbol{h}$, with the resulting transformation given by

$$\boldsymbol{f} = \boldsymbol{h} \circ \mathrm{Pad} \circ \boldsymbol{g} = \tilde{\boldsymbol{h}} \circ \boldsymbol{g}, \tag{1}$$

where $\circ$ denotes composition of functions, Pad denotes adding $D-d$ zeros at the end, and $\tilde{\boldsymbol{h}}$ denotes $\boldsymbol{h} \circ \mathrm{Pad}$. The resulting change of variable formula is more involved (Brehmer and Cranmer, 2020)

$$q_\theta(\boldsymbol{x}) = p(\boldsymbol{u}) \left| \det(\boldsymbol{J}_{\boldsymbol{f}}^\top(\boldsymbol{u})\, \boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{u})) \right|^{-\frac{1}{2}} \Big|_{\boldsymbol{u}=\boldsymbol{f}^{-1}(\boldsymbol{x})}, \tag{2}$$

where $\boldsymbol{J}_{\boldsymbol{f}} \in \mathbb{R}^{D\times d}$. Due to the degeneracy, without good reconstructions the model may learn suboptimal distributions despite having seemingly good log-likelihood estimates (Brehmer and Cranmer, 2020). For this reason, they proposed the Manifold-learning Flow, where $\boldsymbol{h}$ focuses on accurately reconstructing the data and $\boldsymbol{g}$ focuses on learning the density.

While their mechanism is theoretically justified (Loaiza-Ganem et al., 2022), Caterini et al. (2021) noted that the optimization problem becomes challenging, leading to potentially inferior performances in practice. Instead, it can be beneficial to explicitly optimize the KL divergence while penalizing the reconstruction loss. One can define $\tilde{\boldsymbol{h}}^\dagger = \mathrm{Proj} \circ \boldsymbol{h}^{-1}$, where Proj is the operation that takes the $d$ coordinates; this is the proper inverse function of $\tilde{\boldsymbol{h}}$ when $\boldsymbol{x}$ lies precisely on the hypersurface formed by $\tilde{\boldsymbol{h}}$. The resulting loss function is given by

$$\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \left[ -\log q_{\boldsymbol{\theta}}(\boldsymbol{x}) + \lambda \|\boldsymbol{x} - \tilde{\boldsymbol{h}}\left(\tilde{\boldsymbol{h}}^\dagger(\boldsymbol{x})\right)\|^2 \right], \tag{3}$$

where $\lambda > 0$ controls the trade-off between reconstruction and density estimation.

Recall that, to obtain $\log q_{\boldsymbol{\theta}}(\boldsymbol{x})$, we need to compute $\log\det(\boldsymbol{J}_{\boldsymbol{f}}^\top \boldsymbol{J}_{\boldsymbol{f}})$. Caterini et al. (2021) showed that this quantity can be simplified further as

$$\log\det\left(\boldsymbol{J}_{\boldsymbol{f}}^\top \boldsymbol{J}_{\boldsymbol{f}}\right) = 2\log\det\left(\boldsymbol{J}_{\boldsymbol{g}}\right) + \log\det\left(\boldsymbol{J}_{\tilde{\boldsymbol{h}}}^\top \boldsymbol{J}_{\tilde{\boldsymbol{h}}}\right), \tag{4}$$

where $\log\det(\boldsymbol{J}_{\boldsymbol{g}})$ is tractable as $\boldsymbol{g}$ is a square flow, and only $\log\det\left(\boldsymbol{J}_{\tilde{\boldsymbol{h}}}^\top \boldsymbol{J}_{\tilde{\boldsymbol{h}}}\right)$ is a complex term. For optimization we only need the gradient of the log-likelihood. With reasonably small $d$ as in many practical applications, we can compute the $D \times d$ Jacobian matrix exactly using vectorization and forward mode automatic-differentiation with deep learning frameworks, while the cost of obtaining the log-determinant for $d \times d$ is affordable. For larger scale problems, Caterini et al. (2021) proposed an estimator based on Hutchinson trace estimator and conjugate gradients, which can potentially speed up optimizations when the number of conjugate gradient steps is small enough and $d$ is large enough. However, we observed that the exact estimate is largely practical and employed it.

### 2.3 Persistent Homology

In order to measure whether we succeeded in learning and engaging with manifolds with non-trivial topology, we rely on the tool of topological data analysis. We briefly introduce the concept and tools for persistent homology based on Wasserman (2018). Data can reflect the topological structures of the underlying space, and persistent homology provides a principled tool to capture these topological features, often in the form of a persistence diagram. One usually cares about topological features including $H_0$, $H_1$ and $H_2$, where $H_0$ refers to the number of connected components, $H_1$ and $H_2$ refer to the number of one and two-dimensional holes, respectively. A persistence diagram is plotted by monitoring the births and deaths of topological features as balls of varying radii are drawn around data points, with a longer life span implying a more significant feature. A persistence diagram can be generated from a matrix containing pairwise distances. While pairwise Euclidean distances can be used here, the intrinsic distances may be preferred (Fernández et al., 2023). In this work, inspired by the later approach, we use the geodesic distances.

## 3 GEOMETRY OF SINGLE-CHART FLOWS

Largely neglected by previous works concerning normalizing flows on Riemannian manifolds, the flow enables us to learn and utilize the underlying geometric structure of the manifold.

**The Learned Manifold.** A fundamental limitation of a single-chart flow is that it assumes that the manifold is globally diffeomorphic to Euclidean. This is an assumption that is violated by many common manifolds, including sphere and torus. Nevertheless, given a trained normalizing flow, one can reason about the learned Riemannian manifold.

Normalizing flows are by definition bijections, so a flow

acts as a coordinate chart in the Riemannian sense, and naturally induces a well-defined pullback metric in the latent space $\boldsymbol{U}$. We further observe that only $\tilde{\boldsymbol{h}}$ contributes to the manifold embedding, while $\boldsymbol{g}$ is just a reparametrization $\boldsymbol{Z} = \boldsymbol{g}(\boldsymbol{U})$ of the latent space. Hence, we denote $\boldsymbol{z} = \tilde{\boldsymbol{h}}^{\dagger}(\boldsymbol{x})$, and the Riemannian metric in $\boldsymbol{Z}$ takes the form

$$\boldsymbol{G}_{\boldsymbol{Z}}(\boldsymbol{z}) = \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}}\right)^{\top} \boldsymbol{G}_{\boldsymbol{X}}(\boldsymbol{x}) \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}}\right) = \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}}\right)^{\top} \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}}\right), \tag{5}$$

where $\boldsymbol{G}_{\boldsymbol{X}}(\boldsymbol{x}) = \mathbb{I}_D$ is the ambient Euclidean metric. Furthermore, the Jacobian $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \in \mathbb{R}^{D \times d}$ maps a vector from the latent space to a vector in the ambient space as $\boldsymbol{v}_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \boldsymbol{v}_{\boldsymbol{Z}}(\boldsymbol{z})$. In practice, the metric can be obtained through explicitly calculating the Jacobian matrices, while the transformation between the vectors can be obtained through a Jacobian vector product operation, which is reasonably cheap for modern automatic-differentiation systems (Baydin et al., 2018). Further discussions on the geometric interpretation can be found in Section C.2.

As discussed next, the flow enables us to compute geodesics, which are essential for many practical operations on a Riemannian manifold, such as exponential maps and logarithmic maps. We remark that one can in principle obtain other geometric quantities, e.g. the Riemannian curvature, through performing the relevant operations on the learned metric.

**Exponential Maps.** Using the Riemannian metric (5), one can solve the exponential map by solving the geodesic equation as explicitly formulated below.

**Theorem 1.** *The geodesic equation in the latent space induced by the pullback metric* (5) *is given by*

$$\ddot{z}(t)^k = -g^{kl} \sum_{i,j,m} \frac{\partial^2 x_m}{\partial z_i \partial z_j} \frac{\partial x_m}{\partial z_l} \dot{z}^i \dot{z}^j. \tag{6}$$

The proof can be found in Section A. We remark that this is similar to the geodesic induced by Fisher information matrix with Gaussian likelihood (Song et al., 2018). The algorithm for solving exponential maps involves explicitly forming the Jacobian matrix, calculating the Riemannian metric and its inverse, thus has complexity $O(d^3)$. However, $d$ can be much smaller than the dimensionality of the ambient space $D$. Moreover, one can avoid explicitly calculating and storing the higher order gradient tensors. Further discussions can be found in Section C.3.

**Geodesics and Logarithmic Maps.** A geodesic corresponds to a curve with locally minimum energy. One can thus parameterize a curve using a cubic spline in the latent space of the flow and optimize its parameters to minimize the energy as observed in the ambient space (Yang et al., 2018; Detlefsen et al., 2021). Having the optimized curve, we can estimate the distance between the two end points using discretization techniques, and the logarithmic map as the initial velocity.

**Pathology of Single-Chart Flows.** A single-chart flow is, by definition, a bijection between a $d$-dimensional Euclidean space and the learned manifold. As such, the learned manifold is constrained to be diffeomorphic to a Euclidean space. This is a fundamental limitation: when the underlying data manifold is not diffeomorphic to Euclidean, the flow cannot reliably represent its global structure. Perhaps the simplest example is the circle: although it can be parameterized by a single angular coordinate, the fact that $a$ and $a+2\pi$ refer to the same point for any $a \in \mathbb{R}$ means it is not Euclidean. In practice, covering the circle with a single flow inevitably introduces a small "gap" in the learned manifold. As a result, the implied geodesics are necessarily suboptimal as a geodesic needs to traverse the entire circle to connect the two sides of the gap. In addition, the implied topology is wrong.

# 4 MODELING USING MULTI-CHART FLOWS

We resolve the fundamental issue mentioned above, namely, the limitations caused by using a single flow, by considering a mixture of flows that are carefully designed to properly cover the data manifold while respecting its geometric and topological structure. In particular, we aim for each flow to focus on a specific region of the manifold, and we rely on a probabilistic formulation to ensure that overlapping regions exist which is crucial when covering a manifold. We then leverage this construction to extend standard geometric computations, such as geodesics, to this data-driven manifold. We note that the main goal of the proposed generative model is to provide the suitable foundation upon which geometric quantities can be computed, while preserving the intrinsic properties of the manifold.

**Model.** The previous discussions suggest that we need to use multiple flows instead of a single one. We thus consider the following mixture model

$$\sum_{i=1}^{N} \log q_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \sum_{i=1}^{N} \log \left( \sum_{c=1}^{C} q_{\boldsymbol{\theta}}(\boldsymbol{x}_i \,|c) q(c) \right), \tag{7}$$

where $N$ denotes the number of data points, $C \in \mathbb{Z}^+$ denotes the number of charts, and each $q_{\boldsymbol{\theta}}(\boldsymbol{x}\,|c)$ is a

normalizing flow with latent dimensionality $d$ and ambient dimensionality $D$ for chart $c$.

**Training.** With the mixture model formulation, our goal is to minimize the distance between the model distribution and the ground truth distribution. We can write the resulting KL divergence as

$$\mathrm{KL}\left(p(\boldsymbol{x})|q_{\boldsymbol{\theta}}(\boldsymbol{x})\right) = \mathrm{KL}\left(p(\boldsymbol{x}) \middle| \sum_{c=1}^{C} q_{\boldsymbol{\theta}}(\boldsymbol{x}\,|c)q(c)\right). \quad (8)$$

However, as noted by previous works (Brehmer and Cranmer, 2020; Caterini et al., 2021), purely performing Maximum Likelihood Estimation (MLE) training is not sufficient for fitting the degenerate flow to a distribution supported on the manifold. Due to degeneracy, the log-density as calculated by the normalizing flow is in fact the log-density projected onto the learned manifold. Therefore, inspired by Caterini et al. (2021), we calculate the required quantities by adding a regularization term to the log-density of the flow. The resulting log-density for a single data point is given by

$$\log \tilde{q}_{\boldsymbol{\theta}}(\boldsymbol{x}_i\,|c) = \log q_{\boldsymbol{\theta}}(\boldsymbol{x}_i\,|c) - \lambda \|\boldsymbol{x}_i - \tilde{\boldsymbol{h}}_c\left(\tilde{\boldsymbol{h}}_c^{\dagger}(\boldsymbol{x}_i)\right)\|^2. \quad (9)$$

Depending on the problem one may add additional regularization terms here. Upon optimality, one can expect the different flows to learn essentially the same reconstructions in overlapping regions, while modeling the target density through a mixture model. There can be potentially an infinite number of models that fit the manifold and the distribution perfectly.

For training a mixture model, there are generally two commonly employed approaches, i.e. directly performing Maximum Likelihood Estimation (MLE) and using Expectation Maximization (EM). These approaches can be employed in our case as well. In this work, we always employ a uniform prior over $q(c)$, so as to encourage the individual flows to cover approximately equal amount of space and prevent them from not being responsible for any points, leading to a waste of model capacity.

**Maximum Likelihood Estimation (MLE).** Using $\log \tilde{q}_{\boldsymbol{\theta}}(\boldsymbol{x}_i\,|c)$ as in 9, one can define

$$\log \tilde{q}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{i=1}^{N} \log \sum_{c=1}^{C} \exp\left(\log \tilde{q}_{\boldsymbol{\theta}}(\boldsymbol{x}_i\,|c) + \log q(c)\right), \quad (10)$$

which can be implemented using the *logsumexp* operation, enabling direct MLE training of the flows.

**Expectation Maximization (EM).** EM is arguably the most commonly used algorithm for training mixture models, and we adapt the EM algorithm to train multi-chart flows. During EM training, the E-step and the M-step are iteratively applied. In one E-step, the responsibilities of the individual flows can be calculated as

$$r_c(\boldsymbol{x}_i) = \mathrm{stop\_gradient}\left(\mathrm{SoftMax}\left(\log \tilde{q}_{\boldsymbol{\theta}}(\boldsymbol{x}_i\,|c)\right)\right), \quad (11)$$

while in one M-step, one minimizes the loss function

$$L(\boldsymbol{\theta}) = -\sum_{i=1}^{N}\sum_{c=1}^{C} r_c(\boldsymbol{x}_i) \log \tilde{q}_{\boldsymbol{\theta}}(\boldsymbol{x}_i\,|c). \quad (12)$$

Further discussions can be found in Section B. We employ a single gradient descent update for the M-step. One cannot expect a flow to correctly model data points that are far from its responsible area. As such, for an individual flow we only calculate the loss based on points where its responsibilities are above a threshold. The probabilistic formulation using responsibilities allows the charts to overlap in some regions.

**General Strategy.** In preliminary experiments, we did not observe fundamental differences between direct MLE and EM. However, the responsibilities as provided by EM offer interpretability and ways to regularize the flows. In this work, we focus on EM. For training mixture models, one common strategy is to initialize the clusters using simple clustering algorithms like K-Means (Bishop, 2006). Similarly, we apply K-Means to obtain $C$ clusters, training each flow individually within the clusters before running EM.

We remark that training a mixture of normalizing flows has been explored before, where both MLE and EM have been demonstrated to work (Izmailov et al., 2020; Atanov et al., 2020; Ng and Zammit-Mangion, 2023). Nevertheless, it is less investigated as for how to train mixtures of flows when both the data manifold and the distribution are unknown.

## 5 GEOMETRY OF MULTI-CHART FLOWS

**The Learned Manifold.** Similar to the case of classical differential geometry, with multi-chart flows, there may not exist a unique latent space. However, one can still obtain the correct geometric structure from all of them. When the flows are perfectly trained, each individual flow acts as a local chart due to its bijective nature, and we obtain a well-defined atlas that covers the entire manifold by collecting all the charts. However, in practice, we only have access to a finite

and potentially noisy dataset, the modeling and optimization are most likely imperfect, and one *cannot* expect the flows to be perfect.

We argue that we can still reason geometrically through *inconsistent* flows by utilizing a probabilistic treatment, which has been demonstrated useful for learning meaningful geometry in VAEs (Arvanitidis et al., 2018; Hauberg, 2019). Denote the value of interest as $\boldsymbol{e}$. Given $C$ trained flows, since each flow would give us an estimate, we can collect all these estimates to form a set $\{\boldsymbol{e}_c, c = 1, \dots, C\}$, and the task is to assign the right weight to each $\boldsymbol{e}_c$. From a Bayesian perspective, a natural approach to obtain a single estimate $\boldsymbol{e}$ is to use the predictive posterior $\mathbb{E}_{p(c|\,\boldsymbol{x})}[\boldsymbol{e}_c]$.

Interestingly, these weights are precisely the responsibilities given by the EM algorithm. As such, we define the points that lie on the manifold as the set $\{\sum_{c=1,\dots,C} r_c(\boldsymbol{x}')\tilde{\boldsymbol{h}}_c\left(\tilde{\boldsymbol{h}}_c^{\dagger}(\boldsymbol{x}')\right),\ \forall\, \boldsymbol{x}' \in \mathbb{R}^D\}$. This can be seen as a projection of the points $\boldsymbol{x}'$ in the ambient space onto the learned manifold, which is well-defined for sufficiently close points $\boldsymbol{x}'$. We can similarly define the tangent space at point $\boldsymbol{x}$ as the tangent spaces of different charts weighted by $r(\boldsymbol{x})$. When all the flows yield the correct estimate everywhere, these estimates correspond to the ground truth values. Otherwise, the probabilistic treatment results in more robust estimates. In practice, we may set a threshold on the responsibilities and form the predictive only based on those that are above the threshold. Since the flows are trained based on them, we can expect the responsibilities to yield well-defined behavior. Alberti et al. (2024) also observed the benefit of averaging, albeit in the context of Riemannian optimizations.

**Exponential Maps.** Due to the involved definition of manifold structure as discussed above, we need specialized tools to compute the exponential map. Perhaps the most natural way is to apply an algorithm inspired by classical differential geometry, where one solves the exponential map based only on the chart with maximal responsibility, and jumps to another when the most responsible one changes. However, as the responsibilities of the charts decrease, the estimates of the charts can degrade in different ways. Practically, the numerical solvers make it even more challenging. Therefore, we argue that it is beneficial to solve the exponential maps using the responsibilities as calculated in the EM algorithm as weights.

Algorithm 1 provides an Euler-like method. We compute a unique curve directly in the ambient space, and instead of relying exclusively on the exponential map from a single chart, each update step is computed as the weighted average over multiple charts using the corresponding responsibilities. In each responsible chart, we simulate the geodesic for $\Delta t$ time starting from $\boldsymbol{x}_t$ and $\boldsymbol{v}_t$, which can be implemented using any black-box ODE solvers through numerically integrating the geodesic ODE. We provide further ablation studies on the different solvers in Section E.1, verifying that averaging is crucial for a well-behaved solver.

**Algorithm 1:** Algorithm for solving the exponential maps using a fixed $T$.

**for** $t \leftarrow 0$ **to** $T-1$ **do**
  $[\boldsymbol{c}, \boldsymbol{r}] \leftarrow \text{filter}\left(\text{resps}\left(\boldsymbol{x}_t\right), \text{resp\_threshold}\right)$ ; // Get filtered responsibilities
  **for** $c \in \{1, \dots, C\}$ **do**
    $[\boldsymbol{x}_{t+1}^c, \boldsymbol{v}_{t+1}^c] \leftarrow \text{Exp}_{c,\Delta t}\left(\boldsymbol{x}_t, \boldsymbol{v}_t\right)$ ; // Move on the geodesic induced by the $c$-th flow for $\Delta t$
  **end**
  $\boldsymbol{x}_{t+1} \leftarrow \sum_{c=1}^{C} r_c\, \boldsymbol{x}_{t+1}^c$;
  $\boldsymbol{v}_{t+1} \leftarrow \sum_{c=1}^{C} r_c\, \boldsymbol{v}_{t+1}^c$;
**end**

**Geodesics and Logarithmic Maps.** To compute the geodesic and logarithmic map between two points, based on the solution above and Section 3, one might be tempted to parameterize a curve in the latent spaces of all the individual charts. However, a flow can be arbitrarily bad outside its responsible region, and it is unclear how to obtain the final curve based on the ones induced by the charts. Since a geodesic can naturally transition between charts, a more direct approach is to parameterize a curve across multiple latent spaces while accounting for transitions. However, this leads to a particularly challenging optimization problem.

Instead of the costly direct optimization, our key insight is that multi-chart flows enable us to map a curve in the ambient space onto the manifold through the function $\boldsymbol{x} \to \sum_{c=1,\dots,C} r_c(\boldsymbol{x})\tilde{\boldsymbol{h}}_c\left(\tilde{\boldsymbol{h}}_i^{\dagger}(\boldsymbol{x})\right)$. Using this, we propose to parameterize in the ambient space a curve $\gamma_\phi(t)$, e.g., a spline, and optimize the energy of the mapped curve on the learned manifold. The logarithmic map can then be obtained as the initial velocity of the resulting curve (see Algorithm 2).

## 6 EXPERIMENTS

We perform experiments across various datasets to demonstrate the utility of multi-chart flows to model manifold valued data while implicitly capturing the underlying geometry and topology. We compare multi-chart flows (Mult) with single-chart flows trained through both sequential manifold learning and den-

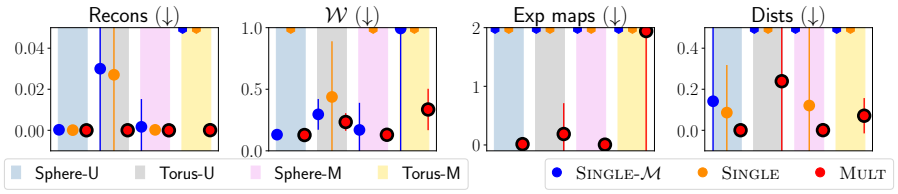


Figure 2: Evaluation metrics on sphere and torus, best methods are circled. Mult consistently leads to better performances. Since geodesics are not unique, in some cases, the found one is not the shortest.

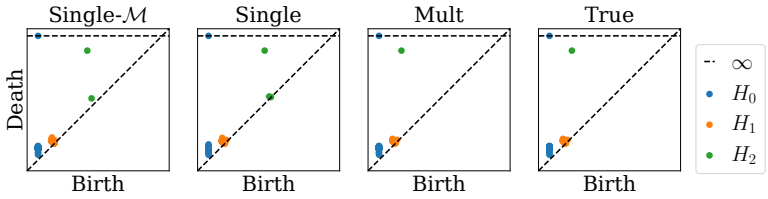


Figure 3: Example persistence diagrams for the Sphere-U data. Points further to the top left corner indicate more significant features. Mult provides the most faithful representation of the underlying topology.

**Algorithm 2:** Algorithm for solving the geodesics and logarithmic maps.

Initialize $\gamma_\phi(t)$;
**for** $i \leftarrow 1$ **to** $N_{iters}$ **do**
  **for** $t \leftarrow 0$ **to** $T-1$ **do**
    $[\boldsymbol{c}, \boldsymbol{r}] \leftarrow \text{filter}\left(\text{resps}\left(\boldsymbol{x}_t\right), \text{resp\_threshold}\right)$ ;
    // Get filtered responsibilities
    $\boldsymbol{x}_t \leftarrow \sum_{c=1}^{C} r_c(\gamma_\phi(t))\tilde{\boldsymbol{h}}_c\left(\tilde{\boldsymbol{h}}_c^{\dagger}\left(\gamma_\phi(t)\right)\right)$;
  **end**
  Minimize $E\left(\{\boldsymbol{x}_t, t = 1, \ldots, T-1\}\right)$ with respect to $\gamma_\phi(t)$
**end**
Fit a new curve $\tilde{\gamma}_\phi(t)$ to $\sum_{c=1}^{C} r_c(\gamma_\phi(t))\tilde{\boldsymbol{h}}_c\left(\tilde{\boldsymbol{h}}_c^{\dagger}\left(\gamma_\phi(t)\right)\right)$;
Compute $\boldsymbol{v}_0$ from $\tilde{\gamma}_\phi(t)$ as the initial velocity;

sity estimation (Single-$\mathcal{M}$) and direct MLE training (Single). For the geodesics, it is known that solutions are not unique; e.g., given two points that are approximately on the opposite of the sphere, the solver may as well find the curve completely opposite of the shortest geodesic, which results in different logarithmic map yet reasonable distance. Hence, we focus on the distances instead of the logarithmic maps. We evaluate the methods using five complementary metrics: (1) the reconstruction loss measured on a test set, (2) the Wasserstein distance of generated samples to a test set, (3)(4) the mean squared errors of the estimated exponential maps and geodesic distances compared with the ground truths, and (5) the faithfulness of the persistence diagram in capturing the topology.

For fair comparisons, we always ensure that all kinds of flows have the same total number of layers for both the outer flows and inner flows. For RealNVP flows the numbers of parameters are exactly the same for single-chart flows and multi-chart flows. For Neural Spline flows since we employ LU Linear Permute before and after every Neural Spline layers, the numbers of parameters differ slightly between single-chart flows and multi-chart flows, but remain comparable. Further experimental details can be found in Section D. We provide code that can be used to reproduce our results at `https://github.com/ksnxr/multi-chart-flows`.

**Sphere and Torus.** We consider modeling distributions supported on two dimensional sphere and two dimensional torus using RealNVP flows (Dinh et al., 2017), where the data and the ground truth geometric quantities are obtained using Geomstats (Miolane et al., 2020, 2024) and Pyro (Bingham et al., 2019). For Single-$\mathcal{M}$ and Single, we use 12 layers for $\boldsymbol{g}$ and 36 layers for $\boldsymbol{h}$. For Mult, we use 4 charts, each chart with 3 layers for $\boldsymbol{g}$ and 9 layers for $\boldsymbol{h}$. For both manifolds, we consider both the uniform distribution, denoted as -U, and the mixture of Von-Mises Fisher and Bivariate Von-Mises distributions, denoted as -M, respectively. Figure 2 shows that Mult yields the best performances in all cases considered. We additionally report the persistence diagrams computed based on the corresponding distance estimates. If these pairwise distances do not accurately reflect the topology of the underlying manifold, the resulting persistent homology may fail to recover the true topological features. Figure 3 verifies that the distances as learned using Mult allow to correctly identify the topological struc-

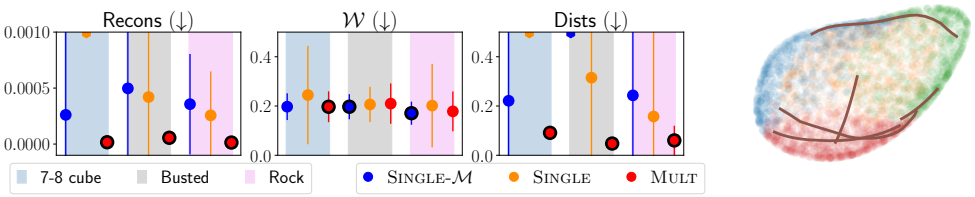


Figure 4: Left: evaluation metrics on triangular meshes, best methods are circled. Mult consistently leads to better reconstructions and distance estimates while having competitive sample quality. Right: a trained multi-chart flow enables us to estimate the logarithmic maps.

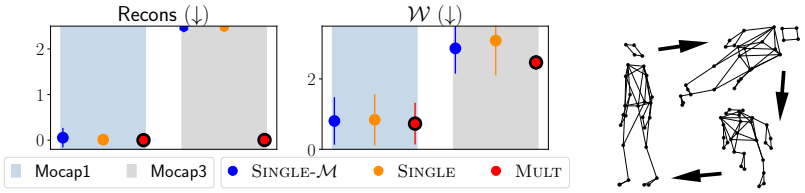


Figure 5: Left: evaluation metrics on Mocap, best methods are circled. Mult has the best reconstructions and sample quality for both Mocap1 and Mocap3. Right: The Mocap frame is rotated, here along the fixed axis. The ambient coordinates thus trace along a low dimensional manifold.

Table 1: The lengths of the 5 segments between 5 points uniformly placed along the topologically-circular data manifold for each method. Ground truth is approximated using the length of the discretized curve with 1000 points. Only Mult correctly reflects the circular nature of the data. Single-chart flows always show a linear-like structure where one segment is significantly larger.

| **Method** | **Lengths** (mean and variance over ten models randomly initialized) | | | | |
|---|---|---|---|---|---|
| Single-$\mathcal{M}$ | $15.1 \pm 10.6$ | $18.5 \pm 13.6$ | $25.4 \pm 17.4$ | $11.9 \pm 0.2$ | $21.7 \pm 15.9$ |
| Single | $18.4 \pm 13.7$ | $25.3 \pm 16.8$ | $11.2 \pm 0.0$ | $15.2 \pm 10.2$ | $21.7 \pm 16.0$ |
| Mult | $11.6 \pm 0.0$ | $11.7 \pm 0.0$ | $11.2 \pm 0.0$ | $11.9 \pm 0.2$ | $11.3 \pm 0.0$ |
| **Ground Truth** | 11.6 | 11.7 | 11.2 | 11.8 | 11.3 |

ture of the sphere; further diagrams can be found in Section E.3.

**Triangular Meshes.** We use the same RealNVP flows (Dinh et al., 2017) as for sphere and torus and triangular meshes provided by Trimesh (Dawson-Haggerty et al., 2025). Inspired by Trimesh, we refine the mesh and approximate the ground truth distances between the points using a neighborhood graph; for a dense enough mesh this approximation is reasonable. Figure 4 shows that for these irregular shapes Mult still provides consistently better reconstructions and distance measures. Also, it enables calculating the exponential maps and logarithmic maps, even if the true ones are intractable.

**Motion Capture Data.** We use the motion capture data from MocapToolbox (Burger and Toiviainen, 2013) and generate datasets where the frame is randomly rotated. We consider (1) the axis is fixed and the angle is uniformly at random which forms a 1-dimensional manifold (*Mocap1*), and (2) both the axis and the angle are uniformly at random, which forms a 3-dimensional manifold due to the connection to the well-known $SO(3)$ manifold (*Mocap3*). As the frame is rotated, it traces a trajectory in the embedding space, which naturally inherits the topology of the employed rotations. The embeddings of the manifold may not be isometric and the ground truth exponential maps and logarithmic maps are intractable. While the *ground truth* distances are in principle intractable, we approximate them numerically by calculating the lengths of discretized curves connecting the embeddings using 1000 points and report them as the ground truth. For a model with correct topology, one would expect the circular topology to be preserved. We employ Neural Spline flows (Durkan et al., 2019). For Mocap1, with Mult we employ 2 charts, each with 6 layers for $\boldsymbol{g}$ and 12 layers for $\boldsymbol{h}$; Single-$\mathcal{M}$ and Single use 12 layers for $\boldsymbol{g}$ and 24 layers for $\boldsymbol{h}$. For Mocap3, Mult

uses 5 charts, each with 4 layers for $\boldsymbol{g}$ and 8 layers for $\boldsymbol{h}$, while SINGLE-$\mathcal{M}$ and SINGLE use 20 layers for $\boldsymbol{g}$ and 40 layers for $\boldsymbol{h}$. Figure 5 shows that MULT always results in the best reconstructions and on-par or better sample quality. Additionally, we analyzed the distances as learned by the models in terms of Mocap1. Table 1 shows that only MULT consistently learn a reasonable measure of lengths, agreeing well with the ground truth and reflecting the correct circular structure of Mocap1.

# 7 DISCUSSION

An important contribution of our paper is to handle non-overlapping charts as they are bound to happen in machine learning applications. We thus developed the first numerical algorithms, to the best of our knowledge, for computing geodesics on such multi-charted manifolds. We also demonstrated that multiple charts are essential when aiming to capture topology. While this is a long-standing knowledge in mathematics, the topic seems to have been underappreciated in the machine learning community.

For generative modeling, one can make arguments that composing multiple smaller models is better than utilizing a single large model (Du and Kaelbling, 2024). Driven by a different motivation, we make similar findings. It is an interesting question as for whether the success of model composition is in part due to the non-trivial manifold structure of real world datasets.

We focus on normalizing flows which enable fast and exact density evaluations, due to their ability to achieve fast inference and efficient computations of geometric quantities involving the Jacobians, along with the tractability of training degenerate normalizing flows. Flow matching (Lipman et al., 2023) is a recent family of algorithms that learns a continuous normalizing flow parameterized by a vector field, without needing numerical integrations during training. It has been extended to Riemannian manifolds (Chen and Lipman, 2024), however both the base distribution and the velocities are defined on the target manifold itself. Free-form flows (Draxler et al., 2024; Sorrenson et al., 2024) is another family of normalizing flow method that has been extended to Riemannian manifolds. However, they are not guaranteed to be bijections, making the differential geometric interpretations less grounded. While we assume that the dimensionality is known beforehand, Zhang et al. (2023) shows that is possible to identify the intrinsic dimensionality while training the flow. We leave further explorations utilizing these methods as future work.

**Limitations.** Multi-chart flows are more technically involved and can be difficult to train especially for manifolds with high curvature. Similarly, computing geodesics becomes more challenging as curvature increases. However, our proposed methodology is, in principle, generic as more advanced flow models can be easily incorporated to improve the performance.

**Conclusion.** Single-chart flows are fundamentally incapable of respecting complex geometric structures. In contrast, we showed that this is possible when using multi-chart flows which can capture the geometry and topology of the underlying space while learning the distribution. In addition, we provided specifically designed numerical methods to approximate geometric quantities, such as geodesics, on the learned manifolds induced by the flows that may not perfectly align. Overall, our work enables computational differential geometry on data manifolds with non-trivial topology.

### Acknowledgements

HY, MH and AK were supported by the Research Council of Finland Flagship programme: Finnish Center for Artificial Intelligence FCAI and by the grants 363317 and 348952, and acknowledge the research environment provided by ELLIS Institute Finland. SH was supported by a research grant (42062) from VILLUM FONDEN, received funding from the European Research Council (ERC) under the European Union's Horizon research and innovation programme (grant agreement 101125993) and was partly funded by the Novo Nordisk Foundation through the Center for Basic Machine Learning Research in Life Science (NNF20OC0062606). GA was supported by the DFF Sapere Aude Starting Grant "GADL". The authors wish to acknowledge CSC - IT Center for Science, Finland, for computational resources.

### References

Acosta, F., Sanborn, S., Duc, K. D., Madhav, M., and Miolane, N. (2023). Quantifying Extrinsic Curvature in Neural Manifolds. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 610–619.

Alberti, G. S., Hertrich, J., Santacesaria, M., and Sciutto, S. (2024). Manifold Learning by Mixture Models of VAEs for Inverse Problems. *Journal of Machine Learning Research*, 25(202):1–35.

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., Hirsh, B., Huang, S., Kalambarkar, K., Kirsch, L., Lazos, M.,

Lezcano, M., Liang, Y., Liang, J., Lu, Y., Luk, C., Maher, B., Pan, Y., Puhrsch, C., Reso, M., Saroufim, M., Siraichi, M. Y., Suk, H., Suo, M., Tillet, P., Wang, E., Wang, X., Wen, W., Zhang, S., Zhao, X., Zhou, K., Zou, R., Mathews, A., Chanan, G., Wu, P., and Chintala, S. (2024). PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM.

Arvanitidis, G., Hansen, L. K., and Hauberg, S. (2018). Latent Space Oddity: on the Curvature of Deep Generative Models. In *International Conference on Learning Representations*.

Atanov, A., Volokhova, A., Ashukha, A., Sosnovik, I., and Vetrov, D. (2020). Semi-Conditional Normalizing Flows for Semi-Supervised Learning. _eprint: 1905.00505.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018). Automatic Differentiation in Machine Learning: a Survey. *Journal of Machine Learning Research*, 18(153):1–43.

Bennett, K., Bradley, P., and Demiriz, A. (2000). Constrained K-Means Clustering. Technical Report MSR-TR-2000-65.

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P. A., Horsfall, P., and Goodman, N. D. (2019). Pyro: Deep Universal Probabilistic Programming. *J. Mach. Learn. Res.*, 20:28:1–28:6.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Information Science and Statistics. Spinger Science+Busines Media, LLC, New York, New York, USA.

Brehmer, J. and Cranmer, K. (2020). Flows for simultaneous manifold learning and density estimation. In *Advances in Neural Information Processing Systems*, volume 33, pages 442–453. Curran Associates, Inc.

Brinkman, E. (2025). Fibonacci Lattice.

Burger, B. and Toiviainen, P. (2013). MoCap Toolbox – A Matlab toolbox for computational analysis of movement data. In Bresin, R., editor, *Proceedings of the 10th Sound and Music Computing Conference*, pages 172–178, Stockholm, Sweden. KTH Royal Institute of Technology.

Caterini, A. L., Loaiza-Ganem, G., Pleiss, G., and Cunningham, J. P. (2021). Rectangular Flows for Manifold Learning. In *Advances in Neural Information Processing Systems*.

Chen, R. T. Q. and Lipman, Y. (2024). Flow Matching on General Geometries. In *The Twelfth International Conference on Learning Representations*.

Dawson-Haggerty et al. (2025). trimesh.

Detlefsen, N. S., Pouplin, A., Feldager, C. W., Geng, C., Kalatzis, D., Hauschultz, H., González-Duque, M., Warburg, F., Miani, M., and Hauberg, S. (2021). StochMan. *GitHub. Note: https://github.com/MachineLearningLifeScience/stochman/*.

Diepeveen, W. (2024). Pulling back symmetric Riemannian geometry for data analysis. _eprint: 2403.06612.

Diepeveen, W., Batzolis, G., Shumaylov, Z., and Schönlieb, C.-B. (2025). Score-based Pullback Riemannian Geometry: Extracting the Data Manifold Geometry using Anisotropic Flows. _eprint: 2410.01950.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using Real NVP. In *International Conference on Learning Representations*.

Do Carmo, M. P. (1992). *Riemannian Geometry*. Mathematics. Theory & applications. Birkhäuser.

Dormand, J. R. and Prince, P. J. (1980). A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26.

Draxler, F., Sorrenson, P., Zimmermann, L., Rousselot, A., and Köthe, U. (2024). Free-form Flows: Make Any Architecture a Normalizing Flow. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 2197–2205. PMLR.

Du, Y. and Kaelbling, L. P. (2024). Position: Compositional Generative Modeling: A Single Model is Not All You Need. In *Forty-first International Conference on Machine Learning*.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019). Neural Spline Flows. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Fernández, X., Borghini, E., Mindlin, G., and Groisman, P. (2023). Intrinsic Persistent Homology via Density-based Metric Learning. *Journal of Machine Learning Research*, 24(75):1–42.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T. H., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T.

(2021). POT: Python Optimal Transport. *Journal of Machine Learning Research*, 22(78):1–8.

Flamary, R., Vincent-Cuaz, C., Courty, N., Gramfort, A., Kachaiev, O., Quang Tran, H., David, L., Bonet, C., Cassereau, N., Gnassounou, T., Tanguy, E., Delon, J., Collas, A., Mazelet, S., Chapel, L., Kerdoncuff, T., Yu, X., Feickert, M., Krzakala, P., Liu, T., and Fernandes Montesuma, E. (2024). POT Python Optimal Transport (version 0.9.5).

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring Network Structure, Dynamics, and Function using NetworkX. In Varoquaux, G., Vaught, T., and Millman, J., editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Fernández del Río, J., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585:357–362.

Hauberg, S. (2019). Only Bayes should learn a manifold (on the estimation of differential geometric structure from data). _eprint: 1806.04994.

Hoffman, M., Sountsov, P., Dillon, J. V., Langmore, I., Tran, D., and Vasudevan, S. (2019). NeuTra-lizing Bad Geometry in Hamiltonian Monte Carlo Using Neural Transport. _eprint: 1903.03704.

Houdouin, P., Ollila, E., and Pascal, F. (2023). Regularized EM Algorithm. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Izmailov, P., Kirichenko, P., Finzi, M., and Wilson, A. G. (2020). Semi-Supervised Learning with Normalizing Flows. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4615–4630. PMLR.

Kalatzis, D., Ye, J. Z., Pouplin, A., Wohlert, J., and Hauberg, S. (2022). Density estimation on smooth manifolds with normalizing flows. _eprint: 2106.03500.

Kruiff, F. d., Bekkers, E., Öktem, O., Schönlieb, C.-B., and Diepeveen, W. (2024). Pullback Flow Matching on Data Manifolds. _eprint: 2410.04543.

Lee, John M. (2018). *Introduction to Riemannian Manifolds*. Graduate Texts in Mathematics. Springer International Publishing AG, Cham, Switzerland, 2nd edition.

Levy-Kramer, J. (2018). k-means-constrained.

Li, H., Zhang, K., and Jiang, T. (2005). The regularized EM algorithm. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, pages 807–812. AAAI Press. Place: Pittsburgh, Pennsylvania.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. (2023). Flow Matching for Generative Modeling. In *The Eleventh International Conference on Learning Representations*.

Loaiza-Ganem, G., Ross, B. L., Cresswell, J. C., and Caterini, A. L. (2022). Diagnosing and Fixing Manifold Overfitting in Deep Generative Models. *Transactions on Machine Learning Research*.

Loaiza-Ganem, G., Ross, B. L., Hosseinzadeh, R., Caterini, A. L., and Cresswell, J. C. (2024). Deep Generative Models through the Lens of the Manifold Hypothesis: A Survey and New Connections. *Transactions on Machine Learning Research*.

Miolane, N., Guigui, N., Brigant, A. L., Mathe, J., Hou, B., Thanwerdas, Y., Heyder, S., Peltre, O., Koep, N., Zaatiti, H., Hajri, H., Cabanes, Y., Gerald, T., Chauchat, P., Shewmake, C., Brooks, D., Kainz, B., Donnat, C., Holmes, S., and Pennec, X. (2020). Geomstats: A Python Package for Riemannian Geometry in Machine Learning. *Journal of Machine Learning Research*, 21(223):1–9.

Miolane, N., Pereira, L. F., Utpala, S., Guigui, N., Brigant, A. L., Hzaatiti, Cabanes, Y., Mathe, J., Koep, N., elodiemaignant, ythanwerdas, xpennec, tgeral68, Christian, Nguyen, T. M., Peltre, O., Harvey, J., pchauchat, julesdeschamps, Barthélemy, Q., mortenapedersen, Maya95assal, Abdellaoui-Souhail, Myers, A., Ambellan, F., Florent-Michel, Heyder, S., Talbar, S., Mont-Marin, Y. d., and Marius (2024). geomstats/geomstats: Geomstats v2.8.0.

Moor, M., Horn, M., Rieck, B., and Borgwardt, K. (2020). Topological Autoencoders. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7045–7054. PMLR.

Murphy, K. P. (2023). *Probabilistic Machine Learning: Advanced Topics*. MIT Press.

Nathaniel Saul, C. T. (2019). Scikit-TDA: Topological Data Analysis for Python.

Ng, T. L. J. and Zammit-Mangion, A. (2023). Mixture Modeling with Normalizing Flows for Spherical Density Estimation. _eprint: 2301.06404.

Nishimoto, T. (2007). On the Lusternik–Schnirelmann category of Stiefel manifolds. *Topology and its Applications*, 154(9):1956–1960.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22(57):1–64.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pennec, X. (2006). Intrinsic Statistics on Riemannian Manifolds: Basic Tools for Geometric Measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127–154.

Rezende, D. J., Papamakarios, G., Racaniere, S., Albergo, M., Kanwar, G., Shanahan, P., and Cranmer, K. (2020). Normalizing flows on tori and spheres. In III, H. D. and Singh, A., editors, *Proceedings of the 37th international conference on machine learning*, volume 119 of *Proceedings of machine learning research*, pages 8083–8092. PMLR.

Ross, B. L., Loaiza-Ganem, G., Caterini, A. L., and Cresswell, J. C. (2024). Neural Implicit Manifold Learning for Topology-Aware Density Estimation. *Transactions on Machine Learning Research*.

Rozo, L., González-Duque, M., Jaquier, N., and Hauberg, S. (2025). Riemann\ˆ2\: Learning Riemannian Submanifolds from Riemannian Data. In *The 28th International Conference on Artificial Intelligence and Statistics*.

Schonsheck, S., Chen, J., and Lai, R. (2020). Chart Auto-Encoders for Manifold Structured Data. _eprint: 1912.10094.

Shampine, L. F. (1986). Some Practical Runge-Kutta Formulas. *Mathematics of Computation*, 46(173):135–150. Publisher: American Mathematical Society.

Sidheekh, S., Dock, C. B., Jain, T., Balan, R., and Singh, M. K. (2022). VQ-Flows: Vector quantized local normalizing flows. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 1835–1845. PMLR.

Song, Y., Song, J., and Ermon, S. (2018). Accelerating Natural Gradient with Higher-Order Invariance. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4713–4722. PMLR.

Sorrenson, P., Draxler, F., Rousselot, A., Hummerich, S., Zimmermann, L., and Koethe, U. (2024). Lifting Architectural Constraints of Injective Flows. In *The Twelfth International Conference on Learning Representations*.

Stimper, V., Liu, D., Campbell, A., Berenz, V., Ryll, L., Schölkopf, B., and Hernández-Lobato, J. M. (2023). normflows: A PyTorch Package for Normalizing Flows. *Journal of Open Source Software*, 8(86):5361.

Stimper, V., Schölkopf, B., and Miguel Hernandez-Lobato, J. (2022). Resampling Base Distributions of Normalizing Flows. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 4915–4936. PMLR.

Sun, X., Liao, D., MacDonald, K., Zhang, Y., Huguet, G., Wolf, G., Adelstein, I., Rudner, T. G. J., and Krishnaswamy, S. (2025). Geometry-Aware Generative Autoencoder for Warped Riemannian Metric Learning and Generative Modeling on Data Manifolds. In *The 28th International Conference on Artificial Intelligence and Statistics*.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Wasserman, L. (2018). Topological Data Analysis. *Annual Review of Statistics and Its Application*, 5(Volume 5, 2018):501–532. Publisher: Annual Reviews Type: Journal Article.

Yang, T., Arvanitidis, G., Fu, D., Li, X., and Hauberg, S. (2018). Geodesic clustering in deep generative models. In *arXiv preprint*.

Zhang, M., Sun, Y., Zhang, C., and Mcdonagh, S. (2023). Spread Flows for Manifold Modelling. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 11435–11456. PMLR.

## Checklist

1. For all models and algorithms presented, check if you include:

(a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]

(b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]

(c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:

(a) Statements of the full set of assumptions of all theoretical results. [Yes]

(b) Complete proofs of all theoretical results. [Yes]

(c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:

(a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]

(b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]

(c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

(a) Citations of the creator If your work uses existing assets. [Yes]

(b) The license information of the assets, if applicable. [Yes]

(c) New assets either in the supplemental material or as a URL, if applicable. [Yes]

(d) Information about consent from data providers/curators. [Not Applicable]

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

# Learning Geometry and Topology via Multi-Chart Flows: Supplementary Materials

## A PROOF OF THEOREM 1

We restate Theorem 1, and provide the proof.

**Theorem 2.** *The geodesic equation in the latent space induced by the pullback metric is given by*

$$\ddot{z}(t)^k = -g^{kl} \sum_{i,j,m} \frac{\partial^2 x_m}{\partial z_i \partial z_j} \frac{\partial x_m}{\partial z_l} \dot{z}^i \dot{z}^j. \tag{13}$$

*Proof.* Recall that the pullback metric is given by

$$\boldsymbol{g} = \left( \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \right)^\top \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}}, \tag{14}$$

$$g_{ij} = \sum_m \frac{\partial x_m}{\partial z_i} \frac{\partial x_m}{\partial z_j}. \tag{15}$$

The Christoffel symbols and the geodesic equation are given by (Lee, John M., 2018) (Equation 5.8 and Equation 4.16)

$$\Gamma_{ij}^k = \frac{1}{2} g^{kl} \left( \partial_i g_{jl} + \partial_j g_{il} - \partial_l g_{ij} \right), \tag{16}$$

$$\ddot{z}(t)^k + \dot{z}^i(t) \dot{z}^j(t) \Gamma_{ij}^k(z(t)) = 0. \tag{17}$$

We have

$$\partial_l g_{ij} = \sum_m \frac{\partial^2 x_m}{\partial z_i \partial z_l} \frac{\partial x_m}{\partial z_j} + \frac{\partial x_m}{\partial z_i} \frac{\partial^2 x_m}{\partial z_j \partial z_l}, \tag{18}$$

and

$$\frac{1}{2} \left( \partial_i g_{jl} + \partial_j g_{il} - \partial_l g_{ij} \right) \tag{19}$$

$$= \frac{1}{2} \left( \sum_m \frac{\partial^2 x_m}{\partial z_j \partial z_i} \frac{\partial x_m}{\partial z_l} + \frac{\partial x_m}{\partial z_j} \frac{\partial^2 x_m}{\partial z_l \partial z_i} + \frac{\partial^2 x_m}{\partial z_i \partial z_j} \frac{\partial x_m}{\partial z_l} + \frac{\partial x_m}{\partial z_i} \frac{\partial^2 x_m}{\partial z_l \partial z_j} \right. \tag{20}$$

$$\left. - \frac{\partial^2 x_m}{\partial z_i \partial z_l} \frac{\partial x_m}{\partial z_j} - \frac{\partial x_m}{\partial z_i} \frac{\partial^2 x_m}{\partial z_j \partial z_l} \right) \tag{21}$$

$$= \frac{1}{2} \left( \sum_m 2 \frac{\partial^2 x_m}{\partial z_i \partial z_j} \frac{\partial x_m}{\partial z_l} \right) = \sum_m \frac{\partial^2 x_m}{\partial z_i \partial z_j} \frac{\partial x_m}{\partial z_l}. \tag{22}$$

As such,

$$\dot{z}^i(t) \dot{z}^j(t) \Gamma_{ij}^k(z(t)) = \dot{z}^i(t) \dot{z}^j(t) \frac{1}{2} g^{kl} \left( \partial_i g_{jl} + \partial_j g_{il} - \partial_l g_{ij} \right) \tag{23}$$

$$= \dot{z}^i(t) \dot{z}^j(t) g^{kl} \sum_m \frac{\partial^2 x_m}{\partial z_i \partial z_j} \frac{\partial x_m}{\partial z_l} = g^{kl} \sum_m \frac{\partial^2 x_m}{\partial z_i \partial z_j} \frac{\partial x_m}{\partial z_l} \dot{z}^i \dot{z}^j, \tag{24}$$

and

$$\ddot{z}(t)^k = -g^{kl} \sum_{i,j,m} \frac{\partial^2 x_m}{\partial z_i \partial z_j} \frac{\partial x_m}{\partial z_l} \dot{z}^i \dot{z}^j. \tag{25}$$

□

The above expression explicitly gives the form of the geodesic accelerations and affords tractable implementations using automatic-differentiation systems, as discussed in detail in Section C.3.

# B EM TRAINING

We briefly explain the entire EM procedure, using as references Murphy (2023) and Bishop (2006). We provide an outline of the general procedure of using the EM algorithm to train the model, and provide the expressions for the precise EM training procedure in our case.

Denote $f(c_i)$ as an arbitrary set of distributions over $c$ for each $\boldsymbol{x}_i$. One can write

$$\sum_{i=1}^{N} \log q(\boldsymbol{x}_i) = \sum_{i=1}^{N} \log \left( \sum_{c_i} q(\boldsymbol{x}_i, c_i) \right) = \sum_{i=1}^{N} \log \left( \sum_{c_i} f(c_i) \frac{q(\boldsymbol{x}_i, c_i)}{f(c_i)} \right). \tag{26}$$

Using Jensen's inequality, we have

$$\sum_{i=1}^{N} \log( \sum_{c_i} f(c_i) \frac{q(\boldsymbol{x}_i, c_i)}{f(c_i)} ) \geq \sum_{i=1}^{N} \sum_{c_i} f(c_i) \log \left( \frac{q(\boldsymbol{x}_i, c_i)}{f(c_i)} \right), \tag{27}$$

which is the Evidence Lower BOund (ELBO). In order to achieve maximum likelihood training, we would like to maximize the above quantity.

For the E step, observe that

$$\sum_{c_i} f(c_i) \log \left( \frac{q(\boldsymbol{x}_i, c_i)}{f(c_i)} \right) = \sum_{c_i} f(c_i) \left( \log \frac{q(c_i | \boldsymbol{x}_i) q(\boldsymbol{x}_i)}{f(c_i)} \right) \tag{28}$$

$$= \sum_{c_i} f(c_i) \left( \log \frac{q(c_i | \boldsymbol{x}_i)}{f(c_i)} \right) + \sum_{c_i} f(c_i) \log q(\boldsymbol{x}_i) = -\mathrm{KL}\left( f(c_i) | q(c_i | \boldsymbol{x}_n) \right) + \log q(\boldsymbol{x}_i). \tag{29}$$

As such, we can set $f(c_i)$ to $q(c_i | \boldsymbol{x}_i)$ to maximize it. Given $q(c_i)$ and $q(\boldsymbol{x}_i | c_i)$, $q(c_i | \boldsymbol{x}_i)$ can be obtained in closed-form using the Bayes rule as

$$q(c_i | \boldsymbol{x}_i) = \frac{q(\boldsymbol{x}_i | c_i) q(c_i)}{q(\boldsymbol{x}_i)} = \frac{q(\boldsymbol{x}_i | c_i) q(c_i)}{\sum_{c=1}^{C} q(\boldsymbol{x}_i | c) q(c)}. \tag{30}$$

Note that $f(c_i)$ thus has an intuitive interpretation of *responsibilities* of the charts on the point $\boldsymbol{x}_i$. We thus refer to it accordingly, and denote it as $r_c(\boldsymbol{x}_i)$.

For the M step, we use $r_c(\boldsymbol{x}_i)$ as derived in the E step and optimize for the best $q(\boldsymbol{x}_i | c)$ (and possibly $q(c)$). In other words, we now need to maximize the following

$$\sum_{i=1}^{N} \sum_{c=1}^{C} r_c(\boldsymbol{x}_i)(\boldsymbol{x}_i) \log \left( q(\boldsymbol{x}_i, c) \right) = \sum_{i=1}^{N} \sum_{c=1}^{C} r_c(\boldsymbol{x}_i) \log q(\boldsymbol{x}_i | c) + \sum_{i=1}^{N} \sum_{c=1}^{C} r_c(\boldsymbol{x}_i) \log q(c). \tag{31}$$

Observe that the optimization for $q(c)$ can be carried out in closed-form, while the optimization for $q(\boldsymbol{x}_i | c)$ affords standard gradient based training.

In the specific case of our multi-chart flows, we directly fix $q(c)$ to be uniform over the charts, and train the individual normalizing flows to model $q(\boldsymbol{x}_i | c)$. However, since we would like to enforce the flows to perform

reconstructions as well, following Caterini et al. (2021) we additionally add a term penalizing the reconstruction error, resulting in

$$\log \tilde{q}_{\boldsymbol{\theta}}(\boldsymbol{x}_i \,|c) = \log q_{\boldsymbol{\theta}}(\boldsymbol{x}_i \,|c) + \lambda \|\boldsymbol{x}_i - \tilde{\boldsymbol{h}_c}\left(\tilde{\boldsymbol{h}_c}^{\dagger}(\boldsymbol{x}_i)\right)\|^2. \tag{32}$$

As noted by Brehmer and Cranmer (2020), depending on the problem, one can add additional terms, e.g. a term that penalizes the hidden variables $\tilde{\boldsymbol{h}_c}^{\dagger}(\boldsymbol{x}_i)$.

To summarize, when training the mixture model using EM, we alternate between the E step, which sets $r_c(\boldsymbol{x}_i)$ based on the flows' log-density evaluations, and the M step, which trains each individual flow based on its responsibilities of the data points.

Additionally, sometimes it is beneficial to include a term penalizing the deviation of the mean responsibilities of the charts averaged over the data points. Since we set $q(c)$ to be uniform over the charts, we expect the mean responsibilities to be close to uniform as well especially when the batch size is larger. We therefore add a loss term in the form of

$$\lambda \sum_{c=1}^{C} \left( \frac{\sum_{i=1}^{N} r_i}{N} - \frac{1}{N} \right)^2. \tag{33}$$

We remark that it has been demonstrated useful to add regularization terms in EM in general (Li et al., 2005; Houdouin et al., 2023).

### B.1 EM and direct MLE

Both EM and direct MLE are in principle applicable for training multi-chart flows, and we could also focus on MLE and it works roughly as well. We do not see the specific choice of the training strategy as a main contribution or a key choice, which is exactly why we described both alternatives. In our work, we prefer EM primarily because it directly provides the responsibilities, an interpretable quantity that can be be used to monitor training and develop practical algorithms for solving for the exponential maps and logarithmic maps. Furthermore, it enables the regularizer given by Equation (33). This regularizer encourages the different charts to contribute equally to the data points, and helps to prevent one chart having essentially zero responsible data points, a pathology that may appear when training unregularized models. We provide ablation studies on models trained using EM and direct MLE in Section E.9, verifying that they yield similar performances.

## C GEOMETRY

### C.1 Explanations Concerning the Geometry

Consider a $d$ dimensional surface $\boldsymbol{S}$ that is parameterized by a map $\boldsymbol{f} : \boldsymbol{Z} \subset \mathbb{R}^d \to \mathbb{R}^D$, where $D$ is the ambient dimension and $\boldsymbol{Z}$ is the parameter space. Under proper smoothness conditions, the pullback Riemannian metric $\boldsymbol{G}(\boldsymbol{z}) = \boldsymbol{J}_{\boldsymbol{f}(\boldsymbol{z})}^{\top} \boldsymbol{J}_{\boldsymbol{f}(\boldsymbol{z})} \in \mathbb{R}_{\succ 0}^{d \times d}$ is a positive definite matrix that captures the intrinsic geometry of the surface in the parameter space. For example, it is possible to compute shortest paths in $\boldsymbol{Z}$ while respecting the geometry of $\boldsymbol{S}$. When the topology of $\boldsymbol{S}$ is more intrincate, i.e. not homeomorphic to Euclidean, we need multiple maps, where each of them parameterizes a neighborhood on the surface. Similarly, in our model, each normalizing flow induces a Riemannian metric in the associated latent space.

### C.2 Details on the Geometric Operations

Consider $\tilde{\boldsymbol{h}}$, $\tilde{\boldsymbol{h}}^{\dagger}$, $\boldsymbol{x}$ and $\boldsymbol{z}$ as defined in the main paper. For points that lie precisely on the learned manifold, $\tilde{\boldsymbol{h}}$ and $\tilde{\boldsymbol{h}}^{\dagger}$ are a pair of bijective functions that are inverses of each other.

Given a value of $\boldsymbol{z}$, one can obtain the unique corresponding $\boldsymbol{x} = \tilde{\boldsymbol{h}}(\boldsymbol{z})$. The pullback metric can also be tractably evaluated as $\boldsymbol{J}_{\tilde{\boldsymbol{h}}}^{\top} \boldsymbol{J}_{\tilde{\boldsymbol{h}}}$. Given $\boldsymbol{v}_{\boldsymbol{z}}$, one can also obtain the corresponding $\boldsymbol{v}_{\boldsymbol{x}}$ using the formula $\boldsymbol{v}_{\boldsymbol{x}} = \frac{\partial \tilde{\boldsymbol{h}}}{\partial \boldsymbol{z}} \boldsymbol{v}_{\boldsymbol{z}}$. One way to see that is to note that $\boldsymbol{v}_{\boldsymbol{x}} = \frac{\partial \boldsymbol{x}}{\partial t} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial t} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \boldsymbol{v}_{\boldsymbol{z}}$. These operations are generally well-defined.

However, due to the degeneracy, multiple $\boldsymbol{x}$ can correspond to the same $\boldsymbol{z}$. $\tilde{\boldsymbol{h}}^{\dagger}$ acts as a projection operation that projects data points onto the manifold, thus naturally induces quotient structures. When $\boldsymbol{x}$ lies exactly

on the learned manifold and $\boldsymbol{v_x}$ lies exactly on the tangent space, one can obtain the corresponding $\boldsymbol{v_z}$ using $\boldsymbol{v_z} = \frac{\partial \tilde{\boldsymbol{h}}^\dagger}{\partial \boldsymbol{x}} \boldsymbol{v_x}$. Due to the way that $\tilde{\boldsymbol{h}}^\dagger$ is composed, we have

$$\frac{\partial \tilde{\boldsymbol{h}}^\dagger}{\partial \boldsymbol{x}} = \frac{\partial \left(\text{Proj} \cdot \boldsymbol{h}^{-1}\right)}{\partial \boldsymbol{x}} = \frac{\partial \text{Proj}}{\partial \boldsymbol{h}^{-1}(\boldsymbol{x})} \frac{\partial \boldsymbol{h}^{-1}}{\partial \boldsymbol{x}}. \tag{34}$$

Projection is a linear operation by definition, and its Jacobian matrix is simply given by a rectangular matrix with ones at the entries where the coordinates are retained. As such, when two distinct values $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ correspond to the same $\boldsymbol{z}$, the Jacobian matrices are the same when the corresponding rows of $\frac{\partial \boldsymbol{h}^{-1}}{\partial \boldsymbol{x}_1}$ and $\frac{\partial \boldsymbol{h}^{-1}}{\partial \boldsymbol{x}_2}$ are the same.

In the case of MULT, the above formulas also enable the definition of a transition operator that moves between charts. Specifically, given the latent coordinates of a point in the $i$th chart $\boldsymbol{z}_i$, its corresponding representation in the $j$th chart is given by $\boldsymbol{z}_j = \tilde{\boldsymbol{h}}_j^\dagger \cdot \tilde{\boldsymbol{h}}_i(\boldsymbol{z}_i)$.

### C.3 Implementing the Geodesic Equation

Here we discuss how to implement an efficient and numerically stable version of the geodesic equation using modern automatic-differentiation (AD) systems.

Recall the form of the geodesic equation; at each step we need to compute $-g^{kl} \sum_{i,j,m} \frac{\partial^2 x_m}{\partial z_i \partial z_j} \frac{\partial x_m}{\partial z_l} \dot{z}^i \dot{z}^j$. One can divide it into three parts: the inverse of the Riemannian metric $g^{kl}$, the Jacobian $\frac{\partial x_m}{\partial z_l}$ and the quadratic form $\frac{\partial^2 x_m}{\partial z_i \partial z_j} \dot{z}^i \dot{z}^j$. Noting that the Riemannian metric is precisely given by the outer product of the Jacobian, we only need to obtain the Jacobian and the quadratic form using AD.

**Jacobian.** In general, for a function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$, the cost to obtain its Jacobian is $kn$ for forward mode AD and $km$ for reverse mode AD, where k is some constant dependent on the function (Baydin et al., 2018). In our case, we need to obtain the Jacobian of $\boldsymbol{h}$, which maps from $\mathbb{R}^d$ to $\mathbb{R}^D$, with $d < D$. As such, one may prefer forward mode AD. In practice, the batched Jacobian matrices can be implemented by a vectorization of the function that obtains the individual Jacobian over the batch, where for each sample we obtain the Jacobian matrix using forward mode AD.

**Quadratic Form.** Forward mode AD step efficiently calculates Jacobian-vector product (Baydin et al., 2018). As such, one can first calculate $\frac{\partial x_m}{\partial z_i} \dot{z}^i$, and then calculate $\frac{\partial^2 x_m}{\partial z_i \partial z_j} \dot{z}^i \dot{z}^j$, employing Jacobian-vector product in each. This is to some extent known in the AD community; see e.g. `https://github.com/jax-ml/jax/discussions/8456#discussioncomment-1586157`.

We remark that we do not claim to have the most efficient implementation, as the focus of the paper is on obtaining the correct geometric quantities instead of having the fastest speed in doing so.

### C.4 Complexities of Solving for Geometric Quantities

**Exponential Maps** For SINGLE-$\mathcal{M}$ and SINGLE, solving the exponential maps directly corresponds to solving for the geodesic equation as an initial value problem. In general, this operation has complexity $O(d^3)$, due to the matrix inversion, and each step involves computing the Jacobian matrices of the flows. However, $d$ is generally small, and the Jacobians can be computed reasonably efficiently, e.g. in PyTorch (Ansel et al., 2024), via vectorized operations.

A crude analysis of the computational complexity of MULT roughly involves solving $C$ such exponential maps for SINGLE and SINGLE, since it needs to average over the predictions generated by the individual charts. However, for each position the number of responsible charts can be small in practice, and it is largely efficient.

**Geodesics and Logarithmic Maps** For SINGLE-$\mathcal{M}$ and SINGLE, obtaining the geodesics and logarithmic maps amounts to optimizing the spline in $d$ dimensional latent space, where in each step the flows need to map points from the latent space to the ambient space.

MULT involves more computations, due to the need of optimizing a spline in the ambient $D$ dimensional space. In each step one needs to reconstruct the spline using the model making use of responsibilities.

### C.5 Connections to Gaussian Mixture Models (GMMs)

A Gaussian Mixture Model (GMM) constructs a mixture of Gaussian distributions, which can be expressive when the number of mixture components is large enough. The mixture formulation is similar to our proposed mixture of flows formulation. However, in our case each individual component is a normalizing flow, which is naturally more flexible than a Gaussian distribution. In general, in our scenario there are two ways to employ a GMM for our purpose, 1. using a GMM in the ambient space and 2. using a GMM directly on the Riemannian manifold.

For the first interpretation, a GMM approximates the data density by fitting multivariate normal distributions to the data, and it is ideal if the data indeed follows such a structure. However, when data possesses a nonlinear manifold structure, a standard GMM is suboptimal. For instance, when data is uniformly distributed on a sphere, the GMM may have trouble capturing the nonlinear structure of the underlying manifold, even when the number of mixture components is large. First, since the dimensionality of the data manifold is 2, the covariances of each individual component must be low-rank. Second, our main goal is not to approximate the data density, but instead to learn a parameterization of the underlying Riemannian manifold. We can think of each component of the GMM as a linear map $\boldsymbol{f}_i(\boldsymbol{x}) = \boldsymbol{\mu}_i + \boldsymbol{A}_i\,\boldsymbol{z}$, where $\boldsymbol{A}_i$ is the decomposition of the covariance $\boldsymbol{\Sigma}_i$ and $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_2)$. Clearly, this parameterization is not able to capture the nonlinear geometry of the sphere.

Instead, when using our approach on the same example, motivated by differential geometry, local neighborhoods of the sphere are parameterized using nonlinear bijective functions $\boldsymbol{f}_i : \mathbb{R}^2 \to \mathbb{R}^3$. In order to learn such local parameterizations from data, we leverage rectangular normalizing flows. Unlike Gaussians, these functions can learn the nonlinear structure of the underying manifold. In order to learn the functions, we rely on a probabilistic approach because this allows the model to learn how to separate the local neighborhoods without explicit supervision, through adapting the responsibilities of the mixture model.

To summarize, our goal is to learn a collection of nonlinear bijections $\boldsymbol{f}_i : \mathbb{R}^d \to \mathbb{R}^D$, where each of them parameterizes a local neighborhood of a nonlinear manifold which resembles the data. We utilize a mixture model to enable the functions to learn the neighborhoods in an unsupervised fashion based on the responsibilities. This can be interpreted as similar to a GMM, where the multivariate normal distributions are replaced by normalizing flows. However, we want to highlight that our main goal is to learn the parameterization of the underlying manifold, instead of performing data density estimation in the ambient space.

In terms of the second interpretation, our method is similar to a GMM defined on a given Riemannian manifold. Indeed, there are extensions of the classic GMMs to nonlinear manifolds, using as building blocks the extensions of Gaussian distributions to Riemannian manifolds (Pennec, 2006). However, in these approaches the underlying Riemannian structure is assumed known beforehand, e.g. we know the data lies on a sphere. Instead, our work focuses on actually learning the unknown underlying Riemmannian manifold from data. With our approach, it is possible to learn the manifold structure, and we also provide the numerical tools that enable one to perform downstream tasks, e.g. statistical analysis. Note that normalizing flows are capable of falling back to GMMs when the target distribution is indeed given by a GMM.

### C.6 Connections to Resampling Base Distributions

Stimper et al. (2022) proposed changing the latent prior while using a single normalizing flow to model the data. This implies that the latent space must have the same dimensionality as the data space, as such, we cannot cover the data manifold in a differential geometric sense.

For example, consider data lying on a sphere. The learned latent prior could also be a sphere, which captures the topology of the data manifold and implies that the trainded normalizing flow will approximately be an identity mapping. As such, the flow will not induce a useful pullback metric for computing geodesics. It might be that we can compute paths that follow high density regions, but these are not necessarily geodesics. In addition, we cannot obtain a 2 dimensional latent space.

To summarize, we believe that Stimper et al. (2022) can serve as a surrogate model or as a pre-processing step for our multi-chart flows formulation.

# D EXPERIMENTAL DETAILS

For open source libraries we provide the license information inside brackets after the citations.

We mainly use PyTorch (Ansel et al., 2024) (custom license), NumPy (Harris et al., 2020) (custom license) and SciPy (Virtanen et al., 2020) (BSD-3 license) for the experiments, with the normalizing flows implementations largely based on Stimper et al. (2023) (MIT license). We use Scikit-TDA (Nathaniel Saul, 2019) (MIT license) to plot the persistence diagrams.

For all models, we tune the learning rates among $[1e-4, 3e-4, 1e-3]$, and tune the reconstruction factors among $[100, 1000, 10000]$. We tune the hyperparameters using a single run for each configuration with seed 1, then for the best configuration run two additional runs using as seeds 2 and 3 and report the results obtained using all three runs. For SINGLE-$\mathcal{M}$, in the first half of all epochs the model is trained solely based on reconstruction errors as induced by $\boldsymbol{h}$, while in the second half of all epochs the model is trained solely based on log-probabilities as induced by $\boldsymbol{g}$.

For SINGLE and MULT, we use early stopping solely in terms of the reconstruction loss; for SINGLE-$\mathcal{M}$, in the manifold learning phase we use early stopping in terms of the reconstruction loss, while in the density estimation phase we use early stopping in terms of $\mathcal{W}$. This may lead to some advantages for SINGLE-$\mathcal{M}$ in terms of $\mathcal{W}$.

For sphere, torus and triangular meshes experiments, we use RealNVP flows (Dinh et al., 2017), where the MLPs use Tanh activation, and *double* data type. We remark that it is important to use a smooth activation function when intending to solve the IVP, due to the need to differentiate through the flow. We sample a total of 12000 data points, and use 25% as the validation set and the remaining as the train set. We train all models for a maximum of 1000 epochs, using a batch size of 256. For MULT, 200 epochs are used for pretraining. The pretraining datasets for the charts are based on K-Means clusters obtained through Scikit-Learn (Pedregosa et al., 2011) (BSD-3-Clause license). We use early stopping with patience 50, starting validation after the model has been trained for 100 epochs in the current stage. We use some separate 10000 data points as the test set.

For SINGLE and MULT, we use as validation loss the reconstruction loss on the val set. For SINGLE-$\mathcal{M}$, we use sequential training, where the first phase uses as validation loss the reconstruction loss, and the second phase uses as validation loss the Wasserstein distance.

For solving the exponential maps, most numerical integrations are carried out using *scipy.integrate.solve_ivp* from SciPy, where we use the default Dopri-5 solver (Dormand and Prince, 1980; Shampine, 1986) with $rtol = 1e-3$ and $atol = 1e-6$. For solving the logarithmic maps, our implementation is largely based on Stochman (Detlefsen et al., 2021) (Apache-2.0 license), where we optimize a parameterized spline using gradient descent.

We use Python Optimal Transport (Flamary et al., 2021, 2024) (MIT license) to calculate the Wasserstein distances between batches of data and batches of samples drawn using the models, where the subsamplings are performed for 5 times and each batch has 1024 samples.

## D.1 Error Bars

For all visualizations where an error bar is shown, the bar is plotted as mean $\pm$ 3 times the standard deviation.

The reported means and standard deviations are obtained using multiple runs with different random seeds. Experimental results as reported in the main paper use 10 independent runs unless otherwise stated, while the experimental results as reported in the Appendix generally use 3 independent runs.

## D.2 Sphere and Torus

We consider four distributions on the two dimensional sphere and the two dimensional torus. Specifically, they are: uniform distribution on the sphere (Sphere-U), mixture of von-Mises Fisher (vMF) distribution on the sphere (Sphere-M), uniform distribution on the torus (Torus-U) and mixture of Bivariate von-Mishes (BvM) distribution on the torus (Torus-M).

The datasets are generated mainly using as tools Geomstats (Miolane et al., 2020, 2024) (MIT license) and Pyro (Bingham et al., 2019) (Apache-2.0 license).

For the mixture of vMF distribution, we use four components, with the means given by

$[\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}], [-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, \frac{1.0}{\sqrt{3}}], [-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}], [\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}]$ and $\kappa = 5$.

For the mixture of BvM distribution, we use four components specified in angular coordinates. The means are given by $[0, 0], [\pi, 0], [0, \pi], [\pi, \pi]$. Each component has concentration 1, with the correlation between the two dimensions 0.

When performing evaluations of the exponential maps, the logarithmic maps and distances, based on preliminary experiments, we observe that it is beneficial to choose the evaluation points as distant to each other as possible to make the resulting persistence diagrams clear. As such, on the sphere, we use Fibonacci lattice as implemented by Brinkman (2025) (MIT license). On the torus, since it is the product manifold formed by two circles, we analytically draw samples on the circles using angular coordinates and form the final samples using Cartesian products. For exponential maps, we use 100 data points, resulting in 4950 evaluations, where we solve 128 maps in a batch. For logarithmic maps and distances we use 225 data points, resulting in 25200 evaluations, where we solve 256 maps in a batch. Note that the ground truth distance of $\boldsymbol{x}_1$ to $\boldsymbol{x}_2$ is naturally the same as the ground truth distance of $\boldsymbol{x}_2$ to $\boldsymbol{x}_1$, and the ground truth distance of $\boldsymbol{x}_1$ to itself is naturally 0. The exponential maps and logarithmic maps are evaluated based on the pairs determined by *tril_indices* function in NumPy and PyTorch with offset $-1$.

### D.3 Triangular Meshes

The triangular meshes experiments are largely based on Trimesh (Dawson-Haggerty et al., 2025) (MIT license). We use *7_8ths_cube.stl*, referred to as 7-8 cube, *busted.STL*, referred to as Busted, and *rock.obj.bz2*, referred to as Rock, as provided by Trimesh (Dawson-Haggerty et al., 2025), while refining the meshes partly to make them smooth and the resulting distance approximations reasonable. Additionally, the meshes are normalized.

For the evaluation of distances, we use 100 data points. Following the reasoning in Section D.2, there are a total of 4950 distance evaluations to be carried out, with each batch solving 256 evaluations. Due to the lack of a principled approach to draw maximally distant samples on the meshes, the data points are simply drawn uniformly. Following Dawson-Haggerty et al. (2025), each *ground truth* distance between two points is approximated as the shortest distance between the closest vertices to the points on the graph induced by the mesh and solved using NetworkX (Hagberg et al., 2008) (3-clause BSD license).

### D.4 Motion Capture Data

We use the initial frame from *dance1* as provided by MocapToolbox (Burger and Toiviainen, 2013) (GNU general public license). We construct two variants of Mocap datasets, *Mocap1* and *Mocap3*.

In Mocap1, the initial frame is randomly rotated around the axis given by $[1/\sqrt{14}, 2/\sqrt{14}, 3/\sqrt{14}]$, with the rotation angle uniformly sampled between 0 and $2\pi$. We use 30000 points as the train set, 10000 points as the val set and 10000 points as the test set. In Mocap3, the initial frame is rotated using a random element from SO(3). We use 50000 points as the train set, 10000 points as the val set and 10000 points as the test set. For both datasets, we standardize the data based on the train set. In Mocap1 the batch size is 256, while for Mocap3 in preliminary experiments we observe that using a larger batch size stabilizes training especially for MULT, so we employ a batch size of 1024.

We use Neural Spline flows (Durkan et al., 2019) with ReLU activation and *float* data type. For pretraining as in MULT, we use Constrained K-Means clustering (Bennett et al., 2000) as implemented in Levy-Kramer (2018) (BSD-3-Clause license) to obtain the clusters, where the clusters are constrained to be between $0.9t$ and $1.1t$, where $t$ is the target size given by dividing the total number of samples by the number of charts. We use early stopping with infinite patience and perform validations throughout the process. For Mocap1, SINGLE is trained for the first 20 epochs solely based on reconstructions, while during pretraining of MULT which has a total of 30 epochs the first 20 epochs are solely based on reconstructions and the following 10 epochs also account for log-probabilities. For Mocap3, the first 50 epochs of SINGLE are solely based on reconstructions, while for MULT among the 70 epochs for pretraining the first 50 are solely based on reconstructions and the later 20 epochs also account for log-probabilities. Following Brehmer and Cranmer (2020), a loss term with weight $1e-3$ is added to constrain the latent representations. For MULT, an additional loss term with weight 10 is added to encourage the mean responsibilities of the charts to be more uniform.

In Mocap1, the ground truth latent space is by definition 1. As such, we perform an experiment where we

**Algorithm 3:** Hard_switch algorithm for solving the exponential maps with a total of $T$ steps. $\text{Exp}_c$ denotes the exponential map induced by the $c$th flow.

**while** $\arg\max(resps(\boldsymbol{x}_t)) = c$ *and* $t < T$ **do**
  $\boldsymbol{x}_{t+1} \leftarrow \text{Exp}_c(\boldsymbol{x}_t, \boldsymbol{v}_t)$;
  $t = t + 1$;
**end**

calculate the pairwise distance between 5 rotations of the original frame whose angles are uniformly distributed, solving one distance in each batch. While it is not clear whether the ground truth distances should be the same for these pairs, we should expect them to reflect the circular nature of the latent space.

### D.5 Compute Resources

We mainly use a compute cluster to perform the experiments. The CPU is Intel Xeon Gold 6230 with 192GB memory, the GPU is Nvidia Volta V100 with 32GB memory.

Generally, for each CPU job we use 4 CPU cores, while each GPU job uses one GPU and 10 CPU cores. For jobs that involved training the flows, for Mocap datasets the jobs are run on GPU, otherwise they were run on CPU. The running times of the jobs may vary, but are generally within several hours. SINGLE-$\mathcal{M}$ generally results in faster training than SINGLE and MULT. Experiments involving evaluating the geometric and topological quantities were run on CPU, whose running times may vary and may need more than a day for those with ill-defined geometries. Some preliminary experiments were run, whose results did not make it to the paper, which naturally took up additional compute resources. The total amount of compute used is thus most likely thousands of CPU hours and hundreds of GPU hours.

## E ADDITIONAL EXPERIMENTAL RESULTS

In the result tables, we use Recons to denote the reconstruction errors, $\mathcal{W}$ to denote the Wasserstein distances, Exps to denote the qualities of exponential maps, Logs to denote the qualitities of logarithmic maps and Dists to denote the qualities of distances.

### E.1 Different Exponential Map Solvers for Multi-Chart Flows

Algorithm 1 in the main paper introduces one algorithm for solving for exponential maps based on multi-chart flows. We refer to that algorithm as `Euler`, and discuss two alternative algorithms, `Hard_switch` and `Ambient`.

**Hard_switch.** The naive algorithm for solving the exponential maps involves following along the geodesic in one chart at a time, and is presented in Algorithm 3. We refer to this algorithm as *Hard_switch*. In terms of numerical implementations, this can be achieved by specifying an event inside an ODE solver that terminates the current integration when the most responsible chart changes and jumps to another chart upon the event happens. With *scipy.integrate.solve_ivp*, we can specify the event in two ways: it can be a *discrete* event checking whether the most responsible chart is the current one or a *continuous* event checking the value of the difference between responsibilities of the current chart and the second most responsible one.

**Ambient.** In the Euler algorithm, there is a hyperparameter $T$ that needs to be tuned. One interesting question to be asked is, what if we take the infinite limit? This question leads to an alternative algorithm where one averages over the velocities and accelerations, so as to directly perform the integrations of the geodesics in the ambient space. We derive the acceleration of geodesics when viewed in the ambient space.

**Theorem 3.** *The acceleration of a geodesic can be expressed in the ambient space as*

$$\boldsymbol{a}_{\boldsymbol{X}} = \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{z}} \boldsymbol{v}_{\boldsymbol{Z}} \boldsymbol{v}_{\boldsymbol{Z}} + \boldsymbol{J} \boldsymbol{a}_{\boldsymbol{Z}}, \tag{35}$$

*where $\boldsymbol{a}_{\boldsymbol{Z}}$ is the acceleration as calculated in Theorem 2.*

**Algorithm 4:** Ambient algorithm for solving the exponential maps with a total of $T$ steps. $\text{Geo}_c$ denotes the function that takes in position and velocity and returns velocity and acceleration of the geodesic induced by the $c$th flow. $\text{Integral}_{\Delta t}$ denotes integrating the ODE for time $\Delta t$.

**for** $t \leftarrow 0$ **to** $T-1$ **do**
  $[\boldsymbol{c}, \boldsymbol{r}] \leftarrow \text{filter}\left(\text{resps}\left(\boldsymbol{x}_t\right), \text{resp\_threshold}\right)$ ; // Get filtered responsibilities
  **for** $c$ *in* $\boldsymbol{c}$ **do**
    $\boldsymbol{v}^c_{t+1}, \boldsymbol{a}^c_{t+1} \leftarrow \text{Geo}_c\left(\boldsymbol{x}_t, \boldsymbol{v}_t\right)$;
  **end**
  $\boldsymbol{v}_{t+1} \leftarrow \sum_{c=1}^{C} r_c\, \boldsymbol{v}^c_{t+1}$, $\boldsymbol{a}_{t+1} \leftarrow \sum_{c=1}^{C} r_c\, \boldsymbol{a}^c_{t+1}$;
  $\boldsymbol{x}_{t+1}, \boldsymbol{v}_{t+1} \leftarrow \text{Integral}_{\Delta t}(\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{v}_{t+1}, \boldsymbol{a}_{t+1})$;
**end**

Table 2: The errors of the exponential map integrators in the form of mean ± std, lower is better. The best are highlighted in bold, and the ones with failed runs are italic. *Hard_switch* is notably worse than the other two.

| Data | Euler | Ambient | Hard_switch |
|---|---|---|---|
| Sphere-U | $9.55{\cdot}10^{-3} \pm 6.31{\cdot}10^{-3}$ | $\mathbf{6.5{\cdot}10^{-3}} \pm 2.73{\cdot}10^{-3}$ | $1.39{\cdot}10^{-1} \pm 2.67{\cdot}10^{-2}$ |
| Torus-U | $7.31{\cdot}10^{-2} \pm 1.26{\cdot}10^{-2}$ | $\mathit{3.5{\cdot}10^{-2}} \pm 3.37{\cdot}10^{-3}$ | $5.69{\cdot}10^{-1} \pm 1.05{\cdot}10^{-1}$ |
| Sphere-M | $1.12{\cdot}10^{-2} \pm 6.33{\cdot}10^{-3}$ | $\mathbf{8.79{\cdot}10^{-3}} \pm 2.44{\cdot}10^{-4}$ | $2.24{\cdot}10^{-1} \pm 4.45{\cdot}10^{-2}$ |
| Torus-M | $\mathbf{1.07{\cdot}10^{0}} \pm 2.39{\cdot}10^{-1}$ | $\mathit{2.14{\cdot}10^{16}} \pm 2.14{\cdot}10^{16}$ | $1.87{\cdot}10^{0} \pm 1.63{\cdot}10^{-1}$ |

*Proof.* Recall that the velocity in the latent space and the velocity in the ambient space are related by $\boldsymbol{v_X} = \boldsymbol{J}\,\boldsymbol{v_Z}$. Taking gradient with respect to $t$ on both sides, we have

$$\boldsymbol{a_X} = \frac{\partial\left(\boldsymbol{J}\,\boldsymbol{v_Z}\right)}{\partial t} = \frac{\partial\,\boldsymbol{J}}{\partial t}\,\boldsymbol{v_Z} + \boldsymbol{J}\,\boldsymbol{a_Z} = \frac{\partial\,\boldsymbol{J}}{\partial\,\boldsymbol{z}}\,\boldsymbol{v_Z}\,\boldsymbol{v_Z} + \boldsymbol{J}\,\boldsymbol{a_Z}\,. \tag{36}$$

□

Interestingly, observe that the first term also appears in Theorem 2. Moreover, after $\boldsymbol{a_Z}$ is calculated, we naturally have access to $\boldsymbol{J}$. As such, the acceleration in the ambient space can be calculated with little extra computational overhead. The resulting algorithm is outlined in Algorithm 4.

**Comparisons.** We compare the *discrete* variant of *Hard_switch*, *Euler* and *Ambient*. The results are shown in Table 2. In practice, we observe that Euler as outlined in Algorithm 1 offers reasonable performances and stabilities. Ambient yields good performances in some cases, but is unstable in some others. As expected, Hard_switch consistently performs worse than Euler.

### E.2 Results on Circle

The evaluation metrics of the different algorithms trained to model the uniform distribution on a circle are shown in Figure 8, and the numerical results can be found in Table 3. We do not observe a qualitative difference between direct MLE and EM. We additionally report example samples generated by the models in Figure 6 and

Table 3: Evaluation metrics on circle in the form of mean ± std, lower is better. The best are highlighted in bold.

| Method | Recons | $\mathcal{W}$ |
|---|---|---|
| Single-$\mathcal{M}$ | $3.65{\cdot}10^{-3} \pm 3.66{\cdot}10^{-3}$ | $6.87{\cdot}10^{-1} \pm 4.3{\cdot}10^{-1}$ |
| Single | $4.68{\cdot}10^{-3} \pm 5.63{\cdot}10^{-3}$ | $1.1{\cdot}10^{-1} \pm 2.49{\cdot}10^{-2}$ |
| Mult-Direct | $7.73{\cdot}10^{-7} \pm 4.32{\cdot}10^{-7}$ | $\mathbf{8.24{\cdot}10^{-2}} \pm 2.88{\cdot}10^{-2}$ |
| Mult | $\mathbf{5.48{\cdot}10^{-7}} \pm 1.2{\cdot}10^{-7}$ | $8.56{\cdot}10^{-2} \pm 2.79{\cdot}10^{-2}$ |

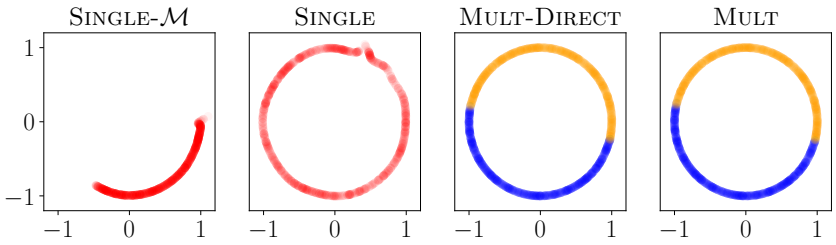


Figure 6: From left to right: SINGLE-$\mathcal{M}$, SINGLE, MULT-DIRECT and MULT. While single-chart flows fail to cover the circle, multi-chart flows are able to provide samples across the circle.

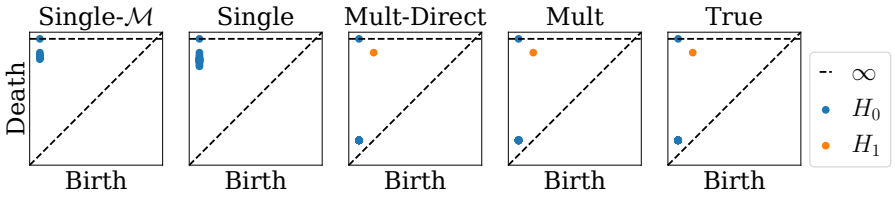


Figure 7: From left to right: example persistence diagrams on SINGLE-$\mathcal{M}$, SINGLE, MULT-DIRECT and MULT. Using a single chart results in pathological failures, where the model fails to cover the entire circle. Only when using multiple charts the models learn the correct topology.

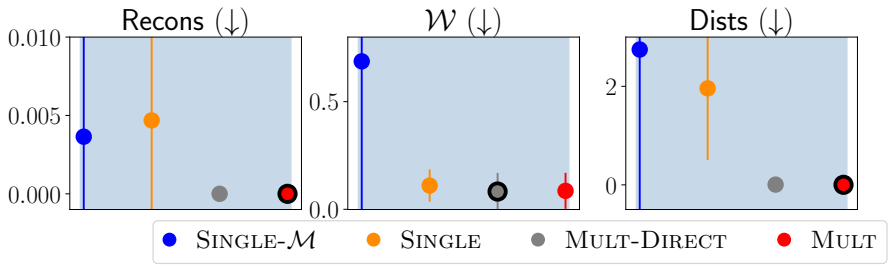


Figure 8: Evaluation metrics of different methods on Circle. Generally, MULT-DIRECT and MULT yield roughly the same level of performances while clearly outperforming SINGLE-$\mathcal{M}$ and SINGLE.

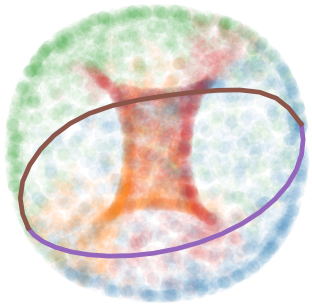

Figure 9: An example of geodesic solver failure on torus. The solver finds the path on the opposite side, resulting in a drastically different logarithmic map and a longer distance.

Table 4: Evaluation metrics concerning modeling of SINGLE-$\mathcal{M}$ on sphere and torus in the form of mean $\pm$ std, lower is better.

| Manifold | Recons | $\mathcal{W}$ |
|---|---|---|
| Sphere-U | $2.49{\cdot}10^{-4} \pm 2.41{\cdot}10^{-4}$ | $1.31{\cdot}10^{-1} \pm 1.28{\cdot}10^{-2}$ |
| Torus-U | $3.0{\cdot}10^{-2} \pm 2.79{\cdot}10^{-2}$ | $2.96{\cdot}10^{-1} \pm 4.16{\cdot}10^{-2}$ |
| Sphere-M | $1.69{\cdot}10^{-3} \pm 4.51{\cdot}10^{-3}$ | $1.7{\cdot}10^{-1} \pm 7.34{\cdot}10^{-2}$ |
| Torus-M | $5.39{\cdot}10^{-2} \pm 4.59{\cdot}10^{-2}$ | $9.93{\cdot}10^{-1} \pm 1.52{\cdot}10^{0}$ |

example persistence diagrams computed based on the models in Figure 7. Both MULT-DIRECT and MULT are able to capture the topology well.

### E.3 Results on Sphere and Torus

We report the numerical results of the different models on sphere and torus in Table 4, Table 5, Table 6, Table 7, Table 8 and Table 9. It is entirely possible for a geodesic solver to return a line that is quite different from the *true* geodesic while still being reasonable, resulting in rather different logarithmic maps and longer distances; Figure 9 shows such an example. We additionally provide example persistence diagrams for Sphere-M, Torus-U and Torus-M in Figure 10; note that example diagrams for Sphere-U were already provided in the main paper. It is clear that MULT consistently provides the most faithful representations of the underlying topology.

### E.4 Results on Triangular Meshes

We report the numerical results of SINGLE-$\mathcal{M}$, SINGLE and MULT on triangular meshes in Table 10, Table 11 and Table 12. We additionally report example persistence diagrams obtained under these settings in Figure 11. Due to the uniform nature of the evaluation points and their relatively small number, the diagrams may not that clearly reflect the underlying topology.

Table 5: Evaluation metrics concerning modeling of SINGLE on sphere and torus in the form of mean $\pm$ std, lower is better.

| Manifold | Recons | $\mathcal{W}$ |
|---|---|---|
| Sphere-U | $8.2{\cdot}10^{-5} \pm 5.42{\cdot}10^{-5}$ | $1.31{\cdot}10^{0} \pm 5.7{\cdot}10^{0}$ |
| Torus-U | $2.7{\cdot}10^{-2} \pm 2.73{\cdot}10^{-2}$ | $4.38{\cdot}10^{-1} \pm 1.51{\cdot}10^{-1}$ |
| Sphere-M | $2.09{\cdot}10^{-4} \pm 2.06{\cdot}10^{-4}$ | $5.21{\cdot}10^{0} \pm 1.86{\cdot}10^{1}$ |
| Torus-M | $5.28{\cdot}10^{-2} \pm 3.36{\cdot}10^{-2}$ | $1.1{\cdot}10^{0} \pm 9.57{\cdot}10^{-1}$ |

Table 6: Evaluation metrics concerning modeling of MULT on sphere and torus in the form of mean ± std, lower is better.

| Manifold | Recons | $\mathcal{W}$ |
|---|---|---|
| Sphere-U | $1.87 \cdot 10^{-6} \pm 5.91 \cdot 10^{-7}$ | $1.28 \cdot 10^{-1} \pm 1.13 \cdot 10^{-2}$ |
| Torus-U | $2.43 \cdot 10^{-5} \pm 7.68 \cdot 10^{-6}$ | $2.34 \cdot 10^{-1} \pm 2.39 \cdot 10^{-2}$ |
| Sphere-M | $1.42 \cdot 10^{-6} \pm 2.44 \cdot 10^{-7}$ | $1.3 \cdot 10^{-1} \pm 1.49 \cdot 10^{-2}$ |
| Torus-M | $6.7 \cdot 10^{-5} \pm 3.83 \cdot 10^{-5}$ | $3.36 \cdot 10^{-1} \pm 5.6 \cdot 10^{-2}$ |

Table 7: Evaluation metrics concerning geometry of SINGLE-$\mathcal{M}$ on sphere and torus in the form of mean ± std, lower is better. The ones with failed runs are italic.

| Manifold | Exps | Logs | Dists |
|---|---|---|---|
| Sphere-U | $\mathit{5.32 \cdot 10^{12}} \pm 1.41 \cdot 10^{13}$ | $1.56 \cdot 10^{0} \pm 1.54 \cdot 10^{0}$ | $1.42 \cdot 10^{-1} \pm 2.86 \cdot 10^{-1}$ |
| Torus-U | $\mathit{4.28 \cdot 10^{18}} \pm 1.13 \cdot 10^{19}$ | $3.04 \cdot 10^{1} \pm 8.44 \cdot 10^{0}$ | $9.74 \cdot 10^{0} \pm 5.2 \cdot 10^{0}$ |
| Sphere-M | $7.55 \cdot 10^{9} \pm 2.26 \cdot 10^{10}$ | $4.58 \cdot 10^{3} \pm 1.37 \cdot 10^{4}$ | $4.57 \cdot 10^{3} \pm 1.37 \cdot 10^{4}$ |
| Torus-M | $\mathit{4.93 \cdot 10^{4}} \pm 1.18 \cdot 10^{5}$ | $3.92 \cdot 10^{2} \pm 6.8 \cdot 10^{2}$ | $3.63 \cdot 10^{2} \pm 6.73 \cdot 10^{2}$ |

Table 8: Evaluation metrics concerning geometry of SINGLE on sphere and torus in the form of mean ± std, lower is better. The ones with failed runs are italic.

| Manifold | Exps | Logs | Dists |
|---|---|---|---|
| Sphere-U | $\mathit{4.25 \cdot 10^{23}} \pm 1.2 \cdot 10^{24}$ | $1.32 \cdot 10^{0} \pm 8.96 \cdot 10^{-1}$ | $8.62 \cdot 10^{-2} \pm 7.73 \cdot 10^{-2}$ |
| Torus-U | $5.54 \cdot 10^{17} \pm 1.66 \cdot 10^{18}$ | $3.06 \cdot 10^{1} \pm 1.46 \cdot 10^{1}$ | $1.05 \cdot 10^{1} \pm 9.45 \cdot 10^{0}$ |
| Sphere-M | $5.46 \cdot 10^{3} \pm 1.61 \cdot 10^{4}$ | $1.63 \cdot 10^{0} \pm 1.18 \cdot 10^{0}$ | $1.21 \cdot 10^{-1} \pm 1.38 \cdot 10^{-1}$ |
| Torus-M | $5.93 \cdot 10^{10} \pm 1.78 \cdot 10^{11}$ | $3.42 \cdot 10^{3} \pm 5.78 \cdot 10^{3}$ | $3.38 \cdot 10^{3} \pm 5.76 \cdot 10^{3}$ |

Table 9: Evaluation metrics concerning geometry of MULT on sphere and torus in the form of mean ± std, lower is better.

| Manifold | Exps | Logs | Dists |
|---|---|---|---|
| Sphere-U | $1.39 \cdot 10^{-2} \pm 1.51 \cdot 10^{-2}$ | $1.11 \cdot 10^{-1} \pm 3.28 \cdot 10^{-2}$ | $3.34 \cdot 10^{-4} \pm 2.21 \cdot 10^{-4}$ |
| Torus-U | $1.87 \cdot 10^{-1} \pm 1.75 \cdot 10^{-1}$ | $4.55 \cdot 10^{0} \pm 1.85 \cdot 10^{0}$ | $2.39 \cdot 10^{-1} \pm 1.82 \cdot 10^{-1}$ |
| Sphere-M | $5.13 \cdot 10^{-3} \pm 4.51 \cdot 10^{-3}$ | $1.4 \cdot 10^{-1} \pm 4.32 \cdot 10^{-2}$ | $6.03 \cdot 10^{-4} \pm 3.16 \cdot 10^{-4}$ |
| Torus-M | $1.94 \cdot 10^{0} \pm 2.13 \cdot 10^{0}$ | $3.19 \cdot 10^{0} \pm 1.05 \cdot 10^{0}$ | $7.1 \cdot 10^{-2} \pm 2.87 \cdot 10^{-2}$ |

Table 10: Evaluation metrics of SINGLE-$\mathcal{M}$ on triangular meshes in the form of mean ± std, lower is better.

| Manifold | Recons | $\mathcal{W}$ | Dists |
|---|---|---|---|
| 7-8 cube | $2.61 \cdot 10^{-4} \pm 2.53 \cdot 10^{-4}$ | $1.97 \cdot 10^{-1} \pm 1.81 \cdot 10^{-2}$ | $2.21 \cdot 10^{-1} \pm 1.57 \cdot 10^{-1}$ |
| Busted | $4.97 \cdot 10^{-4} \pm 1.99 \cdot 10^{-4}$ | $1.97 \cdot 10^{-1} \pm 1.7 \cdot 10^{-2}$ | $5.95 \cdot 10^{-1} \pm 5.17 \cdot 10^{-1}$ |
| Rock | $3.56 \cdot 10^{-4} \pm 1.49 \cdot 10^{-4}$ | $1.71 \cdot 10^{-1} \pm 1.57 \cdot 10^{-2}$ | $2.43 \cdot 10^{-1} \pm 1.23 \cdot 10^{-1}$ |

Table 11: Evaluation metrics of SINGLE on triangular meshes in the form of mean ± std, lower is better.

| Manifold | Recons | $\mathcal{W}$ | Dists |
|---|---|---|---|
| 7-8 cube | $5.69 \cdot 10^{-3} \pm 1.16 \cdot 10^{-2}$ | $2.45 \cdot 10^{-1} \pm 6.65 \cdot 10^{-2}$ | $1.91 \cdot 10^{3} \pm 5.47 \cdot 10^{3}$ |
| Busted | $4.2 \cdot 10^{-4} \pm 2.54 \cdot 10^{-4}$ | $2.06 \cdot 10^{-1} \pm 2.38 \cdot 10^{-2}$ | $3.15 \cdot 10^{-1} \pm 2.13 \cdot 10^{-1}$ |
| Rock | $2.56 \cdot 10^{-4} \pm 1.31 \cdot 10^{-4}$ | $2.02 \cdot 10^{-1} \pm 5.65 \cdot 10^{-2}$ | $1.57 \cdot 10^{-1} \pm 1.5 \cdot 10^{-1}$ |

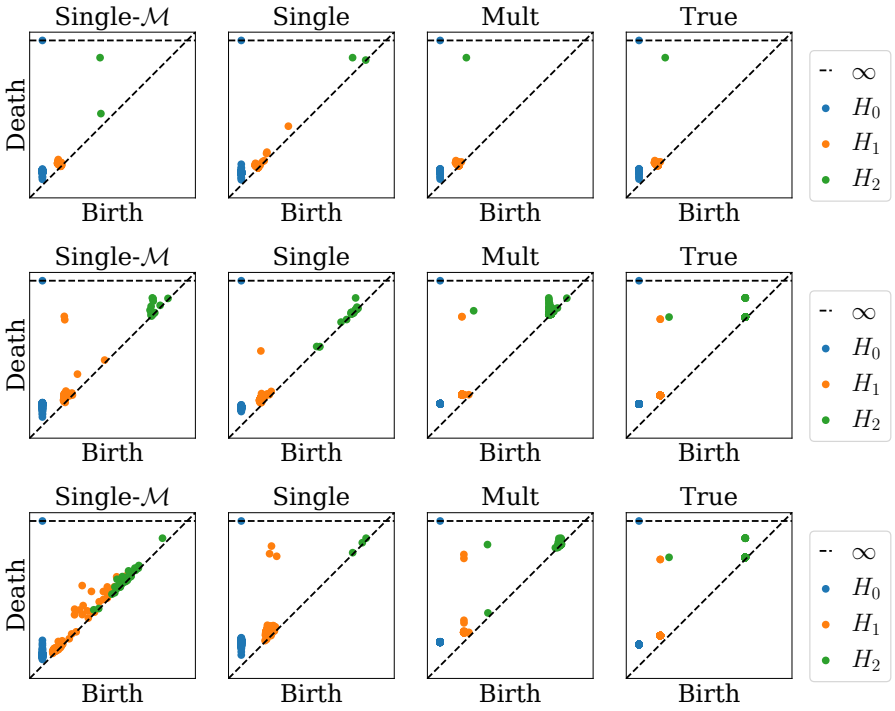


Figure 10: From top to bottom: example persistence diagrams induced by different models on Sphere-M, Torus-U and Torus-M.

Table 12: Evaluation metrics of MULT on triangular meshes in the form of mean ± std, lower is better.

| Manifold | Recons | $\mathcal{W}$ | Dists |
|---|---|---|---|
| 7-8 cube | $1.64{\cdot}10^{-5} \pm 3.37{\cdot}10^{-6}$ | $1.97{\cdot}10^{-1} \pm 2.08{\cdot}10^{-2}$ | $9.11{\cdot}10^{-2} \pm 7.41{\cdot}10^{-3}$ |
| Busted | $5.63{\cdot}10^{-5} \pm 1.63{\cdot}10^{-5}$ | $2.1{\cdot}10^{-1} \pm 2.73{\cdot}10^{-2}$ | $4.76{\cdot}10^{-2} \pm 6.72{\cdot}10^{-3}$ |
| Rock | $1.36{\cdot}10^{-5} \pm 2.94{\cdot}10^{-6}$ | $1.78{\cdot}10^{-1} \pm 2.7{\cdot}10^{-2}$ | $6.03{\cdot}10^{-2} \pm 1.99{\cdot}10^{-2}$ |

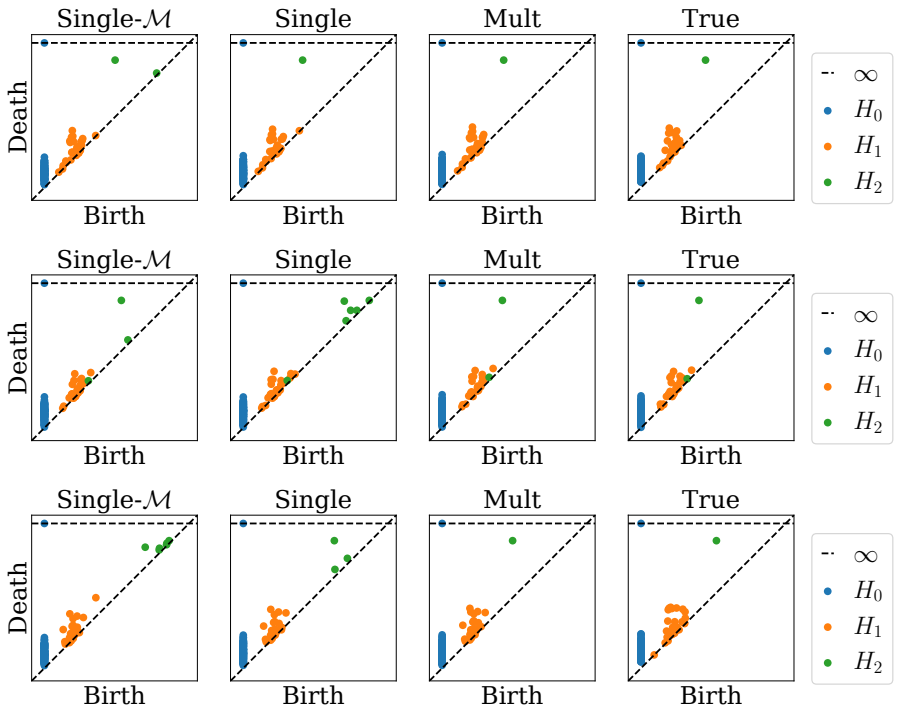


Figure 11: From top to bottom: example persistence diagrams induced by different models on 7-8th cube, Busted and Rock.

Table 13: Evaluation metrics of SINGLE-$\mathcal{M}$ on Mocap in the form of mean ± std, lower is better.

| Manifold | Recons | $\mathcal{W}$ |
|---|---|---|
| Mocap1 | $5.5{\cdot}10^{-2} \pm 7.15{\cdot}10^{-2}$ | $8.05{\cdot}10^{-1} \pm 2.25{\cdot}10^{-1}$ |
| Mocap3 | $3.76{\cdot}10^{0} \pm 2.13{\cdot}10^{0}$ | $2.87{\cdot}10^{0} \pm 2.39{\cdot}10^{-1}$ |

### E.5 Results on Mocap Data

The numerical results of SINGLE-$\mathcal{M}$, SINGLE and MULT on Mocap data are reported in Table 13, Table 14 and Table 15, respectively.

### E.6 Results on Stiefel Manifold

We report results on an additional Riemannian manifold, the Stiefel manifold, which is the manifold formed by the set of all orthonormal $p$ frames in an $n$ dimensional space, based on Geomstats (Miolane et al., 2020, 2024). Note that the Lusternik-Schnirelmann category of Stiefel manifolds are generally known (Nishimoto, 2007), such that we have theoretical guarantees that using a smaller number of charts can cover the manifolds. For MULT, we always use 4 charts. In the following we specify the settings, listing the intrinsic dimensionality $d$, the ambient dimensionality $D$, the number of layers in $\boldsymbol{g}$ num_$\boldsymbol{g}$ and the number of layers in $\boldsymbol{h}$ num_$\boldsymbol{h}$ in each chart:

Table 14: Evaluation metrics of SINGLE on Mocap in the form of mean ± std, lower is better.

| Manifold | Recons | $\mathcal{W}$ |
|---|---|---|
| Mocap1 | $1.14{\cdot}10^{-2} \pm 1.28{\cdot}10^{-2}$ | $8.39{\cdot}10^{-1} \pm 2.44{\cdot}10^{-1}$ |
| Mocap3 | $3.5{\cdot}10^{0} \pm 2.14{\cdot}10^{0}$ | $3.09{\cdot}10^{0} \pm 3.28{\cdot}10^{-1}$ |

Table 15: Evaluation metrics of MULT on Mocap in the form of mean ± std, lower is better.

| Manifold | Recons | $\mathcal{W}$ |
|---|---|---|
| Mocap1 | $6.79 \cdot 10^{-4} \pm 9.64 \cdot 10^{-5}$ | $7.3 \cdot 10^{-1} \pm 1.98 \cdot 10^{-1}$ |
| Mocap3 | $3.97 \cdot 10^{-3} \pm 2.47 \cdot 10^{-4}$ | $2.47 \cdot 10^{0} \pm 7.08 \cdot 10^{-2}$ |

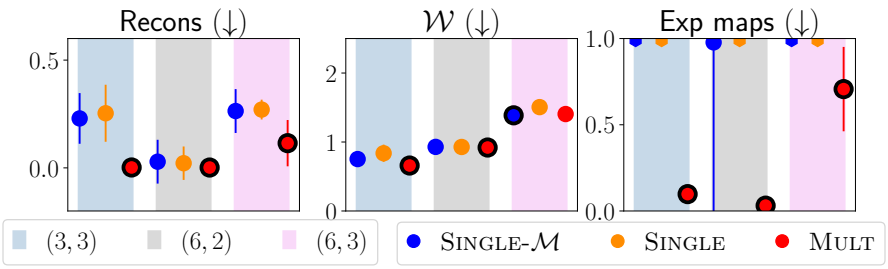


Figure 12: Evaluation metrics of different methods on Stiefel. MULT always yields the best reconstructions and overall the best exponential map solutions, while remaining highly competitive in terms of sample quality.

1. $n = 3, p = 3$: $d = 3$, $D = 9$, SINGLE-$\mathcal{M}$ and SINGLE: num_$\boldsymbol{g}$ = 12, num_$\boldsymbol{h}$ = 36, MULT: num_$\boldsymbol{g}$ = 3, num_$\boldsymbol{h}$ = 9,
2. $n = 6, p = 2$: $d = 9$, $D = 12$, SINGLE-$\mathcal{M}$ and SINGLE: num_$\boldsymbol{g}$ = 24, num_$\boldsymbol{h}$ = 48, MULT: num_$\boldsymbol{g}$ = 6, num_$\boldsymbol{h}$ = 12,
3. $n = 6, p = 3$: $d = 12$, $D = 18$, SINGLE-$\mathcal{M}$ and SINGLE: num_$\boldsymbol{g}$ = 24, num_$\boldsymbol{h}$ = 48, MULT: num_$\boldsymbol{g}$ = 6, num_$\boldsymbol{h}$ = 12.

For the metric induced by ambient Euclidean space, the exponential maps are tractable while the logarithmic maps are not. We thus perform evaluation on the exponential maps, where we sample 320 pairs of initial positions and initial velocities and solve 32 of them in a batch. The initial positions are directly sampled from the data distribution, while the initial velocities are first sampled from the data distribution before projecting onto the tangent spaces corresponding to the initial positions. We observe that SINGLE and SINGLE-$\mathcal{M}$ can lead to challenging integration problems, and we thus set a running time limit of 36 hours. Jobs that exceed the limit are considered failed and the overall result is considered bad.

We report the results in Figure 12. The experimental results of SINGLE-$\mathcal{M}$, SINGLE and MULT are shown in Table 16, Table 17 and Table 18, respectively. We observe that MULT is always the best in terms of reconstructions and yields good exponential map solutions, while being highly competitive and among the best for sample quality.

### E.7 Varying Number of Charts

In the main paper we generally consider the case where there is a single chart or there is a fixed number of charts. Here we perform some additional experiments on the effect of varying number of charts. We always consider the setting where the total number of layers is constant, such that using more charts imply using fewer numbers of layers for the individual flows.

Table 16: Evaluation metrics of SINGLE-$\mathcal{M}$ on Stiefel in the form of mean ± std, lower is better. The ones with failed runs are italic.

| $n, p$ | Recons | $\mathcal{W}$ | Exps |
|---|---|---|---|
| n=3, p=3 | $2.29 \cdot 10^{-1} \pm 3.92 \cdot 10^{-2}$ | $7.53 \cdot 10^{-1} \pm 1.74 \cdot 10^{-2}$ | $\mathit{4.36 \cdot 10^{8}} \pm 3.32 \cdot 10^{8}$ |
| n=6, p=2 | $2.87 \cdot 10^{-2} \pm 3.39 \cdot 10^{-2}$ | $9.29 \cdot 10^{-1} \pm 7.43 \cdot 10^{-3}$ | $9.78 \cdot 10^{-1} \pm 1.21 \cdot 10^{0}$ |
| n=6, p=3 | $2.64 \cdot 10^{-1} \pm 3.39 \cdot 10^{-2}$ | $1.39 \cdot 10^{0} \pm 3.21 \cdot 10^{-3}$ | $\mathit{nan} \pm \mathit{nan}$ |

Table 17: Evaluation metrics of SINGLE on Stiefel in the form of mean ± std, lower is better. The ones with failed runs are italic.

| $n, p$ | Recons | $\mathcal{W}$ | Exps |
|---|---|---|---|
| n=3, p=3 | $2.53 \cdot 10^{-1} \pm 4.41 \cdot 10^{-2}$ | $8.37 \cdot 10^{-1} \pm 4.17 \cdot 10^{-2}$ | $5.56 \cdot 10^{6} \pm 7.86 \cdot 10^{6}$ |
| n=6, p=2 | $2.16 \cdot 10^{-2} \pm 2.59 \cdot 10^{-2}$ | $9.28 \cdot 10^{-1} \pm 2.14 \cdot 10^{-2}$ | $\mathit{2.29 \cdot 10^{-2}} \pm 3.19 \cdot 10^{-3}$ |
| n=6, p=3 | $2.7 \cdot 10^{-1} \pm 1.53 \cdot 10^{-2}$ | $1.51 \cdot 10^{0} \pm 3.66 \cdot 10^{-2}$ | $nan \pm nan$ |

Table 18: Evaluation metrics of MULT on Stiefel in the form of mean ± std, lower is better.

| $n, p$ | Recons | $\mathcal{W}$ | Exps |
|---|---|---|---|
| n=3, p=3 | $1.86 \cdot 10^{-3} \pm 4.72 \cdot 10^{-4}$ | $6.59 \cdot 10^{-1} \pm 3.05 \cdot 10^{-2}$ | $9.78 \cdot 10^{-2} \pm 3.33 \cdot 10^{-3}$ |
| n=6, p=2 | $1.58 \cdot 10^{-3} \pm 3.38 \cdot 10^{-4}$ | $9.2 \cdot 10^{-1} \pm 6.65 \cdot 10^{-3}$ | $3.1 \cdot 10^{-2} \pm 9.22 \cdot 10^{-3}$ |
| n=6, p=3 | $1.15 \cdot 10^{-1} \pm 3.58 \cdot 10^{-2}$ | $1.41 \cdot 10^{0} \pm 4.96 \cdot 10^{-3}$ | $7.07 \cdot 10^{-1} \pm 8.16 \cdot 10^{-2}$ |

We first consider Sphere-U, i.e. the uniform distribution over the two dimensional sphere embedded in three dimensional space. A single chart cannot cover the distribution, while two or more charts are enough. The results for the case where there are 4 charts are directly taken from the main paper. The final result tables are shown in Table 19 and Table 20. In general, the model's performances appear rather stable across the different numbers of charts.

We next consider the case where the target distribution is a part of the uniform distribution over the two dimensional sphere in the three dimensional Euclidean space. In this case, a single chart is already enough. Nevertheless, we consider the case where we use two charts. Since the training objective is based on the entire mixture model, the individual flows are encouraged to cover different areas without being degenerate. As shown in Table 21, using MULT results in highly competitive performances.

In general, as long as the number of charts exceeds what is required to cover the space, it is reasonable to expect that each chart can more easily cover a local region, where the flows cover approximately equal amount of masses. A such, as long as each individual flow has a sufficient number of layers to model part of the target distribution, we can expect the performance to remain stable. In fact, the empirical observation that the performances are stable across the different numbers of layers suggest that our method consistently learns the same underlying manifold, demonstrating stability and robustness.

### E.8 Consistency of the Charts

Our training objective involves the reconstruction losses of the charts at the points where they are reasonably responsible. As such, it can be expected that the different charts yield rather similar reconstructions at the points where they are responsible for. We empirically verify this in Table 22, where we measure the mean squared differences between every pair of reconstructions generated by two flows that have responsibilities over 0.1 for each data point from the test dataset, reported on the sphere and torus datasets that were considered in the main paper. Note that in the main paper we compared the overall reconstructions of the flows with the ground truth, while here we are comparing the reconstructions generated by the individual flows against each

Table 19: Evaluation metrics concerning modeling using varying number of charts on Sphere-U in the form of mean ± std, lower is better. The best are highlighted in bold.

| Num charts | Recons | $\mathcal{W}$ |
|---|---|---|
| 2 | $6.95 \cdot 10^{-6} \pm 1.66 \cdot 10^{-6}$ | $1.33 \cdot 10^{-1} \pm 1.23 \cdot 10^{-2}$ |
| 3 | $3.03 \cdot 10^{-6} \pm 9.82 \cdot 10^{-7}$ | $1.31 \cdot 10^{-1} \pm 1.28 \cdot 10^{-2}$ |
| 4 | $2.22 \cdot 10^{-6} \pm 7.44 \cdot 10^{-7}$ | $\mathbf{1.29 \cdot 10^{-1}} \pm 1.29 \cdot 10^{-2}$ |
| 6 | $\mathbf{1.18 \cdot 10^{-6}} \pm 3.46 \cdot 10^{-7}$ | $1.33 \cdot 10^{-1} \pm 1.53 \cdot 10^{-2}$ |

Table 20: Evaluation metrics concerning geometry using varying number of charts on Sphere-U in the form of mean ± std, lower is better. The best are highlighted in bold.

| Num charts | Exps | Logs | Dists |
|---|---|---|---|
| 2 | $3.11 \cdot 10^{-2} \pm 9.63 \cdot 10^{-3}$ | $8.59 \cdot 10^{-2} \pm 6.7 \cdot 10^{-3}$ | $1.79 \cdot 10^{-4} \pm 2.75 \cdot 10^{-5}$ |
| 3 | $\mathbf{1.66 \cdot 10^{-3}} \pm 9.32 \cdot 10^{-4}$ | $\mathbf{6.56 \cdot 10^{-2}} \pm 1.23 \cdot 10^{-2}$ | $\mathbf{1.58 \cdot 10^{-4}} \pm 2.99 \cdot 10^{-5}$ |
| 4 | $2.28 \cdot 10^{-2} \pm 1.52 \cdot 10^{-2}$ | $1.32 \cdot 10^{-1} \pm 2.86 \cdot 10^{-2}$ | $3.77 \cdot 10^{-4} \pm 1.27 \cdot 10^{-4}$ |
| 6 | $8.91 \cdot 10^{-3} \pm 1.14 \cdot 10^{-2}$ | $1.45 \cdot 10^{-1} \pm 3.63 \cdot 10^{-2}$ | $7.76 \cdot 10^{-4} \pm 2.45 \cdot 10^{-4}$ |

Table 21: Evaluation metrics on a part of Sphere-U in the form of mean ± std, lower is better. The best are highlighted in bold.

| Method | Recons | $\mathcal{W}$ |
|---|---|---|
| SINGLE-$\mathcal{M}$ | $\mathbf{1.04 \cdot 10^{-7}} \pm 3.88 \cdot 10^{-8}$ | $5.55 \cdot 10^{-2} \pm 8.1 \cdot 10^{-3}$ |
| SINGLE | $1.52 \cdot 10^{-6} \pm 4.14 \cdot 10^{-7}$ | $6.21 \cdot 10^{-2} \pm 8.49 \cdot 10^{-3}$ |
| MULT | $1.35 \cdot 10^{-6} \pm 2.85 \cdot 10^{-7}$ | $\mathbf{5.41 \cdot 10^{-2}} \pm 6.64 \cdot 10^{-3}$ |

other. We observe that the inconsistencies among the individual flows are generally small.

### E.9 Direct MLE Compared with EM

In general, direct MLE and EM achieves highly comparable performances. We report experimental results on sphere and torus across 3 independent runs in terms of Recons, $\mathcal{W}$, Exps, Logs and Dists in Table 23, Table 23, Table 25, Table 26 and Table 27.

### E.10 Initialization Strategy for Geodesic Solver

When solving for geodesics by optimizing for the curve, it is clear that the initialization strategy can make a difference. While in the main paper we directly use Stochman's default initialization strategy, below we consider an alternative where we attempt to achieve data-informed initializations. Specifically, we build a K-Nearest Neighbor graph and use the solution based on the graph to initialize the spline. We show the qualities of logarithmic map solutions in Table 28 and qualities of distance solutions in Table 29. Interestingly, the graph-based solution often yields worse results than the naive approach.

### E.11 Computational Time

We report the running times of the different algorithms on sphere and torus across 3 runs in Table 30. MULTOften results in comparable running times as SINGLE.

Table 22: Differences between reconstructions of different charts on boundaries in the form of mean ± std. The differences among the charts are generally small.

| Data | Diffs |
|---|---|
| Sphere-U | $1.15 \cdot 10^{-5} \pm 5.8 \cdot 10^{-6}$ |
| Torus-U | $6.04 \cdot 10^{-5} \pm 8.82 \cdot 10^{-6}$ |
| Sphere-M | $1.41 \cdot 10^{-5} \pm 1.3 \cdot 10^{-6}$ |
| Torus-M | $1.57 \cdot 10^{-4} \pm 3.16 \cdot 10^{-5}$ |

Table 23: Reconstructions of MLE compared with EM across different data.

| Alg | Sphere-U | Torus-U | Sphere-M | Torus-M |
|---|---|---|---|---|
| MLE | $\mathbf{1.68 \cdot 10^{-6}} \pm 4.29 \cdot 10^{-7}$ | $4.65 \cdot 10^{-5} \pm 4.18 \cdot 10^{-5}$ | $1.71 \cdot 10^{-6} \pm 1.58 \cdot 10^{-7}$ | $\mathbf{5.73 \cdot 10^{-5}} \pm 2.62 \cdot 10^{-5}$ |
| EM | $2.22 \cdot 10^{-6} \pm 7.44 \cdot 10^{-7}$ | $\mathbf{2.25 \cdot 10^{-5}} \pm 1.99 \cdot 10^{-6}$ | $\mathbf{1.58 \cdot 10^{-6}} \pm 2.42 \cdot 10^{-7}$ | $6.9 \cdot 10^{-5} \pm 2.96 \cdot 10^{-5}$ |

Table 24: Wasserstein distances of MLE compared with EM across different data.

| Alg | Sphere-U | Torus-U | Sphere-M | Torus-M |
|---|---|---|---|---|
| MLE | $\mathbf{1.29 \cdot 10^{-1}} \pm 1.24 \cdot 10^{-2}$ | $3.26 \cdot 10^{-1} \pm 6.17 \cdot 10^{-2}$ | $\mathbf{1.34 \cdot 10^{-1}} \pm 1.62 \cdot 10^{-2}$ | $\mathbf{3.09 \cdot 10^{-1}} \pm 5.93 \cdot 10^{-2}$ |
| EM | $1.29 \cdot 10^{-1} \pm 1.29 \cdot 10^{-2}$ | $\mathbf{2.46 \cdot 10^{-1}} \pm 2.55 \cdot 10^{-2}$ | $1.34 \cdot 10^{-1} \pm 1.66 \cdot 10^{-2}$ | $3.15 \cdot 10^{-1} \pm 5.93 \cdot 10^{-2}$ |

Table 25: Quality of exponential map solutions of MLE compared with EM across different data.

| Alg | Sphere-U | Torus-U | Sphere-M | Torus-M |
|---|---|---|---|---|
| MLE | $\mathbf{9.05 \cdot 10^{-3}} \pm 7.53 \cdot 10^{-3}$ | $1.07 \cdot 10^{0} \pm 5.73 \cdot 10^{-1}$ | $\mathbf{7.21 \cdot 10^{-3}} \pm 3.68 \cdot 10^{-3}$ | $\mathbf{1.29 \cdot 10^{0}} \pm 2.14 \cdot 10^{-1}$ |
| EM | $2.28 \cdot 10^{-2} \pm 1.52 \cdot 10^{-2}$ | $\mathbf{1.17 \cdot 10^{-1}} \pm 3.99 \cdot 10^{-2}$ | $9.19 \cdot 10^{-3} \pm 5.65 \cdot 10^{-3}$ | $1.5 \cdot 10^{0} \pm 3.82 \cdot 10^{-1}$ |

Table 26: Quality of logarithmic map solutions of MLE compared with EM across different data.

| Alg | Sphere-U | Torus-U | Sphere-M | Torus-M |
|---|---|---|---|---|
| MLE | $1.37 \cdot 10^{-1} \pm 2.59 \cdot 10^{-2}$ | $\mathbf{3.69 \cdot 10^{0}} \pm 1.76 \cdot 10^{0}$ | $\mathbf{1.55 \cdot 10^{-1}} \pm 4.08 \cdot 10^{-2}$ | $\mathbf{2.23 \cdot 10^{0}} \pm 4.87 \cdot 10^{-1}$ |
| EM | $\mathbf{1.32 \cdot 10^{-1}} \pm 2.86 \cdot 10^{-2}$ | $6.41 \cdot 10^{0} \pm 1.02 \cdot 10^{0}$ | $1.67 \cdot 10^{-1} \pm 5.72 \cdot 10^{-2}$ | $2.32 \cdot 10^{0} \pm 6.07 \cdot 10^{-1}$ |

Table 27: Quality of distance solutions of MLE compared with EM across different data.

| Alg | Sphere-U | Torus-U | Sphere-M | Torus-M |
|---|---|---|---|---|
| MLE | $4.3 \cdot 10^{-4} \pm 1.61 \cdot 10^{-4}$ | $\mathbf{1.03 \cdot 10^{-1}} \pm 8.1 \cdot 10^{-2}$ | $8.72 \cdot 10^{-4} \pm 3.16 \cdot 10^{-4}$ | $5.18 \cdot 10^{-2} \pm 1.6 \cdot 10^{-2}$ |
| EM | $\mathbf{3.77 \cdot 10^{-4}} \pm 1.27 \cdot 10^{-4}$ | $4.34 \cdot 10^{-1} \pm 1.38 \cdot 10^{-1}$ | $\mathbf{8.2 \cdot 10^{-4}} \pm 4.1 \cdot 10^{-4}$ | $\mathbf{5.13 \cdot 10^{-2}} \pm 1.71 \cdot 10^{-2}$ |

Table 28: Quality of logarithmic map solutions of graph-based initialization compared with vanilla across different data.

| Data | Raw | Graph |
|---|---|---|
| Sphere-U | $\mathbf{1.32 \cdot 10^{-1}} \pm 2.86 \cdot 10^{-2}$ | $1.6 \cdot 10^{-1} \pm 1.16 \cdot 10^{-2}$ |
| Torus-U | $6.41 \cdot 10^{0} \pm 1.02 \cdot 10^{0}$ | $\mathbf{2.84 \cdot 10^{0}} \pm 9.2 \cdot 10^{-2}$ |
| Sphere-M | $\mathbf{1.67 \cdot 10^{-1}} \pm 5.72 \cdot 10^{-2}$ | $7.86 \cdot 10^{-1} \pm 7.79 \cdot 10^{-2}$ |
| Torus-M | $\mathbf{2.32 \cdot 10^{0}} \pm 6.07 \cdot 10^{-1}$ | $4.76 \cdot 10^{0} \pm 8.04 \cdot 10^{-1}$ |

Table 29: Quality of distance solutions of graph-based initialization compared with vanilla across different data.

| Data | Raw | Graph |
|---|---|---|
| Sphere-U | $\mathbf{3.77 \cdot 10^{-4}} \pm 1.27 \cdot 10^{-4}$ | $3.4 \cdot 10^{-3} \pm 1.82 \cdot 10^{-4}$ |
| Torus-U | $4.34 \cdot 10^{-1} \pm 1.38 \cdot 10^{-1}$ | $\mathbf{4.37 \cdot 10^{-2}} \pm 8.06 \cdot 10^{-3}$ |
| Sphere-M | $\mathbf{8.2 \cdot 10^{-4}} \pm 4.1 \cdot 10^{-4}$ | $3.89 \cdot 10^{-2} \pm 5.9 \cdot 10^{-3}$ |
| Torus-M | $\mathbf{5.13 \cdot 10^{-2}} \pm 1.71 \cdot 10^{-2}$ | $1.27 \cdot 10^{-1} \pm 5.16 \cdot 10^{-2}$ |

Table 30: Computational times of the different algorithms.

| Method | Single-M | Single | Mult |
|---|---|---|---|
| Sphere-U | [2637.04, 1967.51, 1537.88] | [8887.35, 8376.48, 3455.06] | [9924.72, 10077.32, 7130.9] |
| Torus-U | [2972.0, 1033.06, 1417.95] | [5926.32, 3846.62, 8409.61] | [12012.15, 9163.91, 8925.49] |
| Sphere-M | [1622.29, 2792.39, 2560.15] | [12085.48, 9273.44, 4849.75] | [9533.42, 10608.23, 8715.19] |
| Torus-M | [2653.07, 1713.29, 2954.03] | [17894.75, 7220.11, 19395.33] | [12857.94, 15841.65, 11724.81] |

Table 31: Evaluation metrics concerning modeling of SINGLE-$\mathcal{M}$ on data with noise and outliers in the form of mean $\pm$ std, lower is better.

| Data | Recons | $\mathcal{W}$ |
|---|---|---|
| Raw | $8.61 \cdot 10^{-5} \pm 4.9 \cdot 10^{-5}$ | $1.29 \cdot 10^{-1} \pm 1.24 \cdot 10^{-2}$ |
| N 0.01 | $2.26 \cdot 10^{-4} \pm 2.43 \cdot 10^{-4}$ | $1.32 \cdot 10^{-1} \pm 1.18 \cdot 10^{-2}$ |
| N 0.03 | $1.67 \cdot 10^{-4} \pm 1.25 \cdot 10^{-4}$ | $1.3 \cdot 10^{-1} \pm 1.04 \cdot 10^{-2}$ |
| N 0.1 | $3.45 \cdot 10^{-4} \pm 6.62 \cdot 10^{-5}$ | $1.37 \cdot 10^{-1} \pm 1.17 \cdot 10^{-2}$ |
| T 0.01 | $1.18 \cdot 10^{-4} \pm 1.01 \cdot 10^{-4}$ | $1.34 \cdot 10^{-1} \pm 9.25 \cdot 10^{-3}$ |
| T 0.03 | $3.27 \cdot 10^{-4} \pm 2.86 \cdot 10^{-4}$ | $1.33 \cdot 10^{-1} \pm 1.42 \cdot 10^{-2}$ |
| T 0.1 | $4.6 \cdot 10^{-3} \pm 5.96 \cdot 10^{-3}$ | $1.45 \cdot 10^{-1} \pm 1.79 \cdot 10^{-2}$ |

### E.12 Noise and Outliers

We consider the scenario where the data is corrupted by noise and outliers. Specifically, we consider the uniform distribution on two dimensional sphere. We add independent Gaussian or Student-t noise with degrees of freedom 3 to the both the training set and the validation set, while using noiseless test set. We show the results of SINGLE-$\mathcal{M}$, SINGLE and MULT in terms of modeling and geometry in Table 31, Table 32, Table 33, Table 34, Table 35 and Table 36, respectively. The models are still capable of learning the underlying geometry and topology with similar trends of performances, though naturally the performances degrade as the amount of noise and outliers increases.

## F PREVIOUS WORKS

Previous works that study the estimation of geometric structures (Kruiff et al., 2024; Diepeveen et al., 2025; Sun et al., 2025) typically employ a single chart. As such, pathological issues are bound to appear when the method is trained on a dataset that cannot be covered by a single chart. As an illustrative example, we consider Diepeveen et al. (2025). We take their code and train a Riemannian autoencoder, using as data samples drawn from a circle in the two dimensional Euclidean space. A correct Riemannian autoencoder should identify the correct latent dimensionality, i.e. one. However, due to the topology mismatch, their method could not achieve this in practice: the learned variances for the two dimensions are around 0.589 and 0.325, respectively, which

Table 32: Evaluation metrics concerning modeling of SINGLE on data with noise and outliers in the form of mean $\pm$ std, lower is better.

| Data | Recons | $\mathcal{W}$ |
|---|---|---|
| Raw | $6.81 \cdot 10^{-5} \pm 4.66 \cdot 10^{-5}$ | $1.35 \cdot 10^{-1} \pm 9.43 \cdot 10^{-3}$ |
| N 0.01 | $1.8 \cdot 10^{-4} \pm 1.26 \cdot 10^{-4}$ | $6.51 \cdot 10^{1} \pm 2.43 \cdot 10^{2}$ |
| N 0.03 | $1.4 \cdot 10^{-4} \pm 2.97 \cdot 10^{-5}$ | $1.38 \cdot 10^{-1} \pm 1.6 \cdot 10^{-2}$ |
| N 0.1 | $6.83 \cdot 10^{-4} \pm 4.0 \cdot 10^{-4}$ | $1.47 \cdot 10^{-1} \pm 2.32 \cdot 10^{-2}$ |
| T 0.01 | $1.8 \cdot 10^{-4} \pm 1.48 \cdot 10^{-4}$ | $1.34 \cdot 10^{-1} \pm 1.41 \cdot 10^{-2}$ |
| T 0.03 | $1.85 \cdot 10^{-4} \pm 1.08 \cdot 10^{-4}$ | $1.36 \cdot 10^{-1} \pm 1.27 \cdot 10^{-2}$ |
| T 0.1 | $1.11 \cdot 10^{-3} \pm 6.25 \cdot 10^{-4}$ | $1.44 \cdot 10^{-1} \pm 1.33 \cdot 10^{-2}$ |

Table 33: Evaluation metrics concerning modeling of MULT on data with noise and outliers in the form of mean ± std, lower is better.

| Data | Recons | $\mathcal{W}$ |
|---|---|---|
| Raw | $2.22 \cdot 10^{-6} \pm 7.44 \cdot 10^{-7}$ | $1.29 \cdot 10^{-1} \pm 1.29 \cdot 10^{-2}$ |
| N 0.01 | $8.4 \cdot 10^{-6} \pm 9.34 \cdot 10^{-7}$ | $1.36 \cdot 10^{-1} \pm 1.1 \cdot 10^{-2}$ |
| N 0.03 | $1.42 \cdot 10^{-4} \pm 9.29 \cdot 10^{-6}$ | $1.32 \cdot 10^{-1} \pm 8.79 \cdot 10^{-3}$ |
| N 0.1 | $8.55 \cdot 10^{-4} \pm 4.68 \cdot 10^{-5}$ | $1.62 \cdot 10^{-1} \pm 8.89 \cdot 10^{-3}$ |
| T 0.01 | $8.1 \cdot 10^{-6} \pm 1.54 \cdot 10^{-6}$ | $1.32 \cdot 10^{-1} \pm 1.01 \cdot 10^{-2}$ |
| T 0.03 | $1.14 \cdot 10^{-4} \pm 1.95 \cdot 10^{-5}$ | $1.39 \cdot 10^{-1} \pm 1.06 \cdot 10^{-2}$ |
| T 0.1 | $6.97 \cdot 10^{-4} \pm 1.12 \cdot 10^{-4}$ | $1.81 \cdot 10^{-1} \pm 1.2 \cdot 10^{-2}$ |

Table 34: Evaluation metrics concerning geometry of SINGLE-$\mathcal{M}$ on data with noise and outliers in the form of mean ± std, lower is better.

| Data | Exps | Logs | Dists |
|---|---|---|---|
| Raw | $1.86 \cdot 10^{3} \pm 2.62 \cdot 10^{3}$ | $8.58 \cdot 10^{-1} \pm 3.45 \cdot 10^{-1}$ | $2.35 \cdot 10^{-2} \pm 7.66 \cdot 10^{-3}$ |
| N 0.01 | $4.94 \cdot 10^{16} \pm 6.99 \cdot 10^{16}$ | $1.28 \cdot 10^{0} \pm 2.84 \cdot 10^{-1}$ | $8.34 \cdot 10^{-2} \pm 6.0 \cdot 10^{-2}$ |
| N 0.03 | $1.76 \cdot 10^{0} \pm 1.18 \cdot 10^{0}$ | $1.41 \cdot 10^{0} \pm 1.14 \cdot 10^{0}$ | $7.53 \cdot 10^{-2} \pm 9.07 \cdot 10^{-2}$ |
| N 0.1 | $1.19 \cdot 10^{1} \pm 1.17 \cdot 10^{1}$ | $7.02 \cdot 10^{-1} \pm 9.09 \cdot 10^{-2}$ | $2.5 \cdot 10^{-2} \pm 9.8 \cdot 10^{-3}$ |
| T 0.01 | $2.91 \cdot 10^{2} \pm 4.11 \cdot 10^{2}$ | $1.31 \cdot 10^{0} \pm 6.5 \cdot 10^{-1}$ | $1.07 \cdot 10^{-1} \pm 7.74 \cdot 10^{-2}$ |
| T 0.03 | $2.0 \cdot 10^{18} \pm 2.83 \cdot 10^{18}$ | $9.31 \cdot 10^{-1} \pm 2.21 \cdot 10^{-1}$ | $4.4 \cdot 10^{-2} \pm 2.11 \cdot 10^{-2}$ |
| T 0.1 | $4.9 \cdot 10^{-1} \pm 2.23 \cdot 10^{-1}$ | $1.16 \cdot 10^{0} \pm 4.37 \cdot 10^{-1}$ | $6.03 \cdot 10^{-2} \pm 3.76 \cdot 10^{-2}$ |

Table 35: Evaluation metrics concerning geometry of SINGLE on data with noise and outliers in the form of mean ± std, lower is better.

| Data | Exps | Logs | Dists |
|---|---|---|---|
| Raw | $1.91 \cdot 10^{24} \pm 1.91 \cdot 10^{24}$ | $1.09 \cdot 10^{0} \pm 2.0 \cdot 10^{-1}$ | $5.22 \cdot 10^{-2} \pm 6.04 \cdot 10^{-3}$ |
| N 0.01 | $1.43 \cdot 10^{13} \pm 2.02 \cdot 10^{13}$ | $7.86 \cdot 10^{-1} \pm 4.49 \cdot 10^{-2}$ | $3.75 \cdot 10^{-2} \pm 7.46 \cdot 10^{-3}$ |
| N 0.03 | $7.96 \cdot 10^{0} \pm 1.06 \cdot 10^{1}$ | $6.63 \cdot 10^{-1} \pm 4.31 \cdot 10^{-2}$ | $1.89 \cdot 10^{-2} \pm 8.43 \cdot 10^{-3}$ |
| N 0.1 | $1.69 \cdot 10^{1} \pm 2.37 \cdot 10^{1}$ | $7.72 \cdot 10^{-1} \pm 2.98 \cdot 10^{-1}$ | $2.82 \cdot 10^{-2} \pm 1.77 \cdot 10^{-2}$ |
| T 0.01 | $1.51 \cdot 10^{9} \pm 2.09 \cdot 10^{9}$ | $6.9 \cdot 10^{-1} \pm 2.48 \cdot 10^{-1}$ | $2.58 \cdot 10^{-2} \pm 1.45 \cdot 10^{-2}$ |
| T 0.03 | $2.95 \cdot 10^{2} \pm 4.18 \cdot 10^{2}$ | $8.95 \cdot 10^{-1} \pm 3.37 \cdot 10^{-1}$ | $5.87 \cdot 10^{-2} \pm 6.04 \cdot 10^{-2}$ |
| T 0.1 | $7.95 \cdot 10^{1} \pm 1.12 \cdot 10^{2}$ | $1.03 \cdot 10^{0} \pm 7.82 \cdot 10^{-1}$ | $6.96 \cdot 10^{-2} \pm 8.21 \cdot 10^{-2}$ |

Table 36: Evaluation metrics concerning geometry of MULT on data with noise and outliers in the form of mean ± std, lower is better.

| Data | Exps | Logs | Dists |
|---|---|---|---|
| Raw | $2.28 \cdot 10^{-2} \pm 1.52 \cdot 10^{-2}$ | $1.32 \cdot 10^{-1} \pm 2.86 \cdot 10^{-2}$ | $3.77 \cdot 10^{-4} \pm 1.27 \cdot 10^{-4}$ |
| N 0.01 | $3.48 \cdot 10^{-2} \pm 3.55 \cdot 10^{-2}$ | $1.18 \cdot 10^{-1} \pm 3.17 \cdot 10^{-2}$ | $4.71 \cdot 10^{-4} \pm 3.5 \cdot 10^{-4}$ |
| N 0.03 | $1.04 \cdot 10^{-1} \pm 1.42 \cdot 10^{-2}$ | $2.91 \cdot 10^{-1} \pm 2.59 \cdot 10^{-2}$ | $2.56 \cdot 10^{-3} \pm 4.33 \cdot 10^{-4}$ |
| N 0.1 | $1.31 \cdot 10^{-1} \pm 1.42 \cdot 10^{-2}$ | $1.09 \cdot 10^{0} \pm 1.51 \cdot 10^{-2}$ | $1.58 \cdot 10^{-2} \pm 1.42 \cdot 10^{-3}$ |
| T 0.01 | $3.9 \cdot 10^{-2} \pm 4.45 \cdot 10^{-2}$ | $1.09 \cdot 10^{-1} \pm 3.15 \cdot 10^{-2}$ | $4.07 \cdot 10^{-4} \pm 2.1 \cdot 10^{-4}$ |
| T 0.03 | $1.91 \cdot 10^{-1} \pm 3.13 \cdot 10^{-2}$ | $3.09 \cdot 10^{-1} \pm 1.0 \cdot 10^{-2}$ | $1.88 \cdot 10^{-3} \pm 5.07 \cdot 10^{-4}$ |
| T 0.1 | $2.43 \cdot 10^{-1} \pm 2.98 \cdot 10^{-2}$ | $1.26 \cdot 10^{0} \pm 3.25 \cdot 10^{-2}$ | $1.97 \cdot 10^{-2} \pm 2.7 \cdot 10^{-3}$ |

are still large.

Rozo et al. (2025) employed a Bayesian approach, and considered the problem of learning the submanifolds of existing manifolds.