

---

# Multi-chart flows

---

Dimitris Kalatzis\*  
dika@dtu.dk

Johan Ziruo Ye\*  
ziruoYe@gmail.com

Jesper Wohlert\*  
jesper@wohlert.nu

Søren Hauberg\*  
sohau@dtu.dk

## Abstract

We present *Multi-chart flows*, a flow-based model for concurrently learning topologically non-trivial manifolds and statistical densities on them. Current methods focus on manifolds that are topologically Euclidean, enforce strong structural priors on the learned models or use operations that do not scale to high dimensions. In contrast, our model learns the local manifold topology piecewise by “gluing” it back together through a collection of learned coordinate charts. We demonstrate the efficiency of our approach on synthetic data of known manifolds, as well as higher dimensional manifolds of unknown topology, where we show better sample efficiency and competitive or superior performance against current state-of-the-art.

## 1 Introduction

Normalizing flows [25, 29] provide an elegant framework for modelling complex, multimodal probability distributions. Normalizing flows comprise a base distribution  $P_U$  on a latent space  $U$  and a diffeomorphism (a smooth bijection with a smooth inverse), which provides a 1-to-1 mapping of points in the data space to the latent space  $\mathbf{x} = f(\mathbf{u})$  according to this base distribution. The marginal likelihood of a data point can be computed via the change of variables formula  $p(\mathbf{x}) = p(\mathbf{u})|\det J_f(\mathbf{u})|^{-1} = p(\mathbf{u})|\det J_{f^{-1}}(\mathbf{x})|$ . Typically, the base distribution  $P_U$  is chosen to be a standard normal or a uniform distribution, which are defined in Euclidean space.

Real world data, however, often lie on a manifold, with examples including protein structures [2, 14], geological data [18, 28] or graph-structured and hierarchical data [31, 32]. Diffeomorphisms, being bijections between smooth manifolds, preserve the topology of their domain and therefore modelling the density of manifold-valued data is a known failure mode of flows, due to the topological mismatch between the target distribution  $P_{X^*}$  and the base distribution  $P_U$  [6–8]. In response, recent works have constructed flows for specific manifolds, such as tori, spheres and hyperbolic spaces [3, 30].

Still, in many realistic situations one may not know the topology of a given data set a priori, but one may reasonably assume an underlying manifold structure. Such cases generally fall under the *manifold hypothesis* [11], an important heuristic in machine learning, which states that high dimensional data can be described by a low dimensional submanifold embedded in the observation space. So the question then emerges: Can flow models learn the shape of a data manifold along with a probability density function on it? Works that provide an answer to this question can be roughly divided between those that leverage known local geometric information of the manifold [10, 21, 22] and those that learn its topological structure [4]. Our contribution falls in the latter category.

We present *Multi-chart flows*, a model that leverages the class of functions typically learned by flow models to learn a collection of smooth coordinate charts that cover the data manifold. Unlike existing methods, we learn data manifolds with complex (non-Euclidean) topologies, which in turn improves our density estimates (Fig. 1). Furthermore, in contrast to methods that depend on local geometry, our model scales to high dimensional data and is able to achieve competitive or superior performance in all tasks with better sample efficiency and faster runtimes than most of our baselines.

---

\*Section for Cognitive Systems, Technical University of Denmark

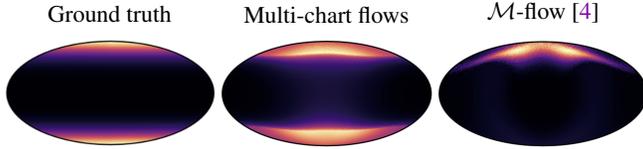


Figure 1: A bimodal distribution on a sphere. Unlike our model, the Euclidean  $\mathcal{M}$ -flow model struggles to push probability mass to cover both modes.

## 2 Topological manifolds and smooth structures

To make subsequent exposition clearer we will briefly review a few basic notions. We begin with the definition of a topological manifold since any smooth manifold is also a topological manifold.

**Definition 1** A topological manifold  $\mathcal{M}$  of dimension  $N$  is locally Euclidean, i.e. each point of  $\mathcal{M}$  has a neighborhood which is homeomorphic to an open subset of  $\mathbb{R}^N$ .

We can now formalize the “locally Euclidean” property of a topological manifold by introducing the notion of *local coordinate charts* on  $\mathcal{M}$ .

**Definition 2** Given an  $N$ -dimensional topological manifold  $\mathcal{M}$ , a coordinate chart on  $\mathcal{M}$  is a pair  $(U, \phi)$ , where  $\phi: U \rightarrow \hat{U}$  is a homeomorphism between the open subsets  $U \subset \mathcal{M}$  and  $\hat{U} \subset \mathbb{R}^N$ .

To have local coordinates for every point on  $\mathcal{M}$  we can define a collection of coordinate charts that covers  $\mathcal{M}$ . This collection is called an atlas and it is denoted by  $\mathcal{A}$ . These definitions are enough<sup>2</sup> to concretely define the topology of a manifold  $\mathcal{M}$ . However, in order to define smooth functions, continuous densities and perform gradient-based optimization on  $\mathcal{M}$ , we need to impose further structure on it, which will allow the use of differential forms. In particular, we need to impose a *smooth structure*.

We first require that our charts be smooth, in the sense that for a function  $f: \mathcal{M} \rightarrow \mathbb{R}$ , the composition  $f \circ \phi^{-1}: \hat{U} \rightarrow \mathbb{R}$  is smooth. Furthermore, given two coordinate charts  $(U_1, \phi_1), (U_2, \phi_2)$  if  $U_1 \cap U_2 \neq \emptyset$ , the composition  $\phi_2 \circ \phi_1^{-1}$  is called the *transition map* between  $\phi_1$  and  $\phi_2$  and is a homeomorphism since it’s composed from homeomorphisms. Two charts are called *smoothly compatible* if their transition map is a diffeomorphism, i.e. a smooth bijection with a smooth inverse (see Fig. 2). If all coordinate charts within an atlas  $\mathcal{A}$  are smoothly compatible, then  $\mathcal{A}$  is called a smooth atlas. If furthermore,  $\mathcal{A}$  contains all smoothly compatible charts then  $\mathcal{A}$  is called *maximal*. We are now ready to give the definition of a smooth structure on a topological manifold.

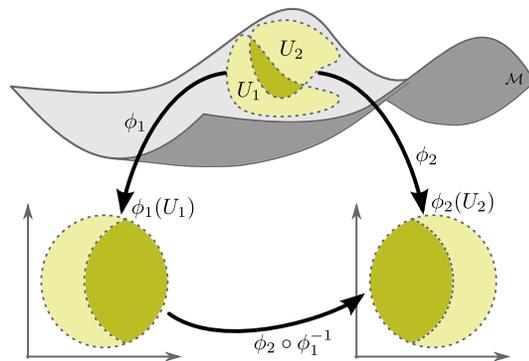


Figure 2: Smoothly compatible charts.

**Definition 3** Given a topological manifold  $\mathcal{M}$ , a smooth structure on  $\mathcal{M}$ , is a maximal smooth atlas.

In practice, when “gluing” together a collection of multiple smoothly compatible charts, it is sufficient to choose the maximal smooth atlas  $\mathcal{A}$  to be the atlas which contains all of these charts.

## 3 Multi-chart Flows

We now present our main contribution, *Multi-chart flows* (MCF). We introduce the model of the generative process we are considering and subsequently discuss details on inference and training.

<sup>2</sup>A topological manifold is also a Hausdorff space and it is second countable but we omit these conditions from the definition as they are not relevant for the setting we are considering.

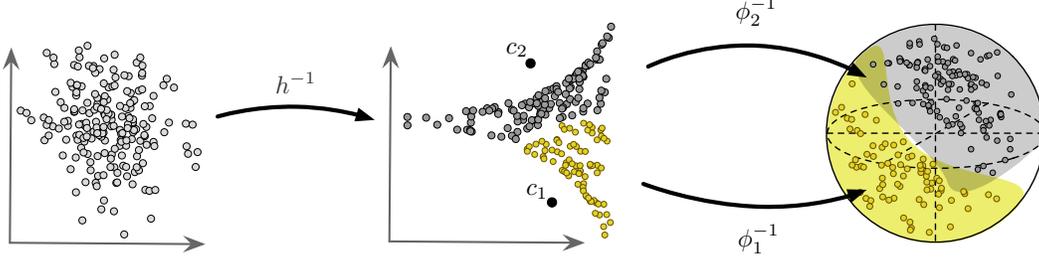


Figure 3: Overview of the generative process proposed in Multi-chart flows. The base distribution on a lower dimensional Euclidean space is transformed by a flow,  $h^{-1}$ . Each point is then mapped onto a manifold embedded in a higher dimensional space through its corresponding coordinate chart based on its distance to the nearest chart center.

### 3.1 Model specification and generative process

We are considering a data manifold  $\mathcal{M}$  of dimension  $d$ , embedded in some Euclidean space  $\mathbb{R}^D$  with  $d \ll D$ . We will think of densities on  $\mathcal{M}$  as arising from transforming points in its coordinate patches,  $\hat{U}_i \subset \mathbb{R}^d$ . These points are themselves a result of a diffeomorphism  $h : \mathbb{R}^d \rightarrow \mathbb{R}^d$ , which transforms a simple base distribution defined in  $\mathbb{R}^d$ , such as a standard Gaussian or a uniform distribution. We furthermore segment  $\mathbb{R}^d$  into  $N$  local neighborhoods  $\hat{U}_i$ . We represent these with points  $\mathbf{c}_i$  in  $\mathbb{R}^d$ , which we consider the centers of  $\hat{U}_i$ . We then “assign” a diffeomorphism to each such neighborhood to construct local coordinate charts  $(\hat{U}_i, \phi_i)$ .

We now turn our attention to the generative process we seek to model. We begin by first sampling from our base distribution  $P_{\tilde{U}}$  and then transform points with  $h^{-1}$ :

$$\mathbf{u} = h^{-1}(\tilde{\mathbf{u}}), \quad \text{where } \tilde{\mathbf{u}} \sim P_{\tilde{U}} \quad (1)$$

We compute pairwise distances with neighborhood centers  $\mathbf{c}_i$  for all transformed points  $\mathbf{u}$ , find the closest neighborhood center for each point and then map it onto the manifold with the corresponding coordinate map  $\phi_i^{-1} : \hat{U}_i \rightarrow \mathcal{M}$ :

$$\mathbf{x} = \phi_k^{-1}(\mathbf{u}), \quad \text{where } k = \arg \min_i \{ \|\mathbf{u} - \mathbf{c}_i\|_2^2 \}_{i=1}^N \text{ and } \mathbf{x} \in \mathcal{M} \subset \mathbb{R}^D. \quad (2)$$

### 3.2 Inference

The inference process for MCF is similar to established flow models. The generative model is inverted in order to map the observed data distribution to the base distribution in latent space. For  $N$  coordinate charts, the defined generative process induces a density on the embedded data manifold:

$$p_{\mathcal{M}}(\mathbf{x}) = \sum_{i=1}^N p(\mathbf{u}) |\det G_i(\mathbf{u})|^{-1/2} = p(\mathbf{u}) |\det G_k(\mathbf{u})|^{-1/2}, \quad (3)$$

where  $p(\mathbf{u})$  is the density in  $U \subset \mathbb{R}^d$  and  $G_i = (J_{\phi_i^{-1}})^\top J_{\phi_i^{-1}}$  the metric tensor induced by the embedding  $\phi_i^{-1}$  with the corresponding Jacobian matrix  $J_{\phi_i^{-1}} \in \mathbb{R}^{d \times D}$ . For ease of computation, we define this density to be 0 when  $i \neq k$  in the sum in eq. 3, where  $k$  is defined in eq. 2. The density in  $U$  is a simple application of the typical change of variables and expands as usual:

$$p(\mathbf{u}) = p(\tilde{\mathbf{u}}) |\det J_{h^{-1}}(\tilde{\mathbf{u}})|^{-1} = p(h(\mathbf{u})) |\det J_h(\mathbf{u})|. \quad (4)$$

Finally the complete log likelihood on the data manifold is given by:

$$\log p_{\mathcal{M}}(\mathbf{x}) = \log p(h(\phi_k(\mathbf{x}))) + \log |\det J_h(\phi_k(\mathbf{x}))| - \frac{1}{2} \log |\det G(\phi_k^{-1}(\mathbf{x}))|. \quad (5)$$

While formally a coordinate chart is defined as a homeo- or diffeomorphic map, we can construct it as an embedding, i.e. an injective map which preserves the topology of its domain and as such, it is invertible with respect to its codomain. With regard to the embedded manifold, our injective maps still function as coordinate charts. To construct such a map we append  $D - d$  zeros to  $\mathbf{u}$  and map to  $\mathbb{R}^D$  with  $\phi^{-1}$ . We denote this “augmented” variable by  $\mathbf{u}' = [u_0, \dots, u_D, 0, \dots, 0]^\top$  and for the remainder of the paper we will use this symbol to refer to this construction.

### 3.3 Introducing a lower bound to the density

While the determinant term in eq. 5 can be computed exactly, it involves evaluating the metric tensor  $G$ , which is prohibitively expensive even for a modest number of dimensions, since computing the determinant is an  $O(d^3)$  operation. We introduce a lower bound to the log likelihood by replacing the determinant with the trace of  $G$  which is an  $O(d)$  operation. A sketch of a proof follows. For a more complete proof see Appendix A. Denoting the singular values of  $J_{\phi^{-1}}$  by  $\{s_i\}_{i=1}^d$  we have:

$$\log p(\mathbf{x}) = \log p(\mathbf{u}) - \frac{1}{2} \log \det |G(\mathbf{u})| = \log p(\mathbf{u}) - \frac{1}{2} \sum_i \log s_i^2. \quad (6)$$

Using Jensen’s inequality we can bound this density by:

$$\log p(\mathbf{x}) \geq \log p(\mathbf{u}) - \frac{1}{2} \log \sum_i s_i^2 = \log p(\mathbf{u}) - \frac{1}{2} \log \text{Tr}[(J_{\phi^{-1}}(\mathbf{u}))^\top J_{\phi^{-1}}(\mathbf{u})]. \quad (7)$$

We can compute the trace efficiently using Hutchinson’s estimator [16] finally arriving at:

$$\log p(\mathbf{x}) \geq \log p(\mathbf{u}) - \frac{1}{2} \log \mathbb{E}_{p(\mathbf{v})} [\|\mathbf{v}^\top J_\phi\|_2^2], \quad \text{with } \mathbf{v} \sim \mathcal{N}(0, I_D). \quad (8)$$

Alternatively, since we construct  $\phi : \mathbb{R}^D \rightarrow \mathcal{M} \subset \mathbb{R}^D$  such that  $\mathbf{u}' \mapsto \phi(\mathbf{u}')$ , we can apply a coarse-grained approximation  $\frac{1}{2} \log \det |G(\mathbf{u})| \approx \log \det |J_{\phi^{-1}}|$  of the transformed volume from  $U$  to  $\mathcal{M}$ .

### 3.4 Training

To compute the likelihood of a data point  $\mathbf{x}$  we must determine which coordinate chart needs to be inverted to map  $\mathbf{x}$  to the latent space  $U$  or inversely, which coordinate chart gave rise to  $\mathbf{x}$ . This information is not readily available, since we segment the latent space instead of the observation space.

One solution to this problem is by having an ‘oracle’, which accurately approximates the inverse of a coordinate chart given a data point. Given a large and flexible enough family of mappings, the chart inverses can be approximated reasonably well for all practical purposes. We use *continuously indexed flows (CIFs)* [6] to act as an ‘oracle map’. Whereas typical flows represent a single diffeomorphism  $\mathbf{x} = f(\mathbf{u})$ , CIFs represent a whole family of indexed diffeomorphisms  $\{F(\mathbf{u}; \mathbf{a})\}_{\mathbf{a} \in A}$ . The indices are continuous and are represented by variables  $\mathbf{a} \in \mathbb{R}^{d_A}$ . We note that a separate unrestricted encoder network should also perform well in this task.

Brehmer and Cranmer [4] showed that for manifold probabilistic models it is desirable to split training into a *manifold learning* phase, where the model learns to reconstruct the manifold, and then a *density learning* phase, where the parameters of the probabilistic model are learned on the converged reconstruction. This increases robustness of training with better convergence rates and local optima. We follow the same approach, where in the reconstruction phase we train only the chart maps, i.e.  $\phi$  and  $\phi^{-1}$  on the following mean squared error loss.

$$\mathcal{L}_R = \frac{1}{M} \sum_{i=1}^M \|\mathbf{x}_i - \phi^{-1}(\phi(\mathbf{x}_i))\|_2^2, \quad (9)$$

where  $M$  is the number of samples in a batch. While in the manifold learning phase we train only the  $h$  transform on the  $\mathbf{u}$  coordinates by maximum likelihood. This way we restrict the evaluation of the negative log likelihood on the learned manifold:

$$\mathcal{L}_M = -\frac{1}{M} \sum_{i=1}^M \log p(\mathbf{u}_i) = -\frac{1}{M} \sum_{i=1}^M \log p(\phi^{-1}(\mathbf{x}_i)) \quad (10)$$

Beyond the advantages mentioned above, further benefits of this scheme include avoiding evaluating the metric tensor during training (since the coordinate maps  $\phi$  are trained on the reconstruction error eq. 9) and also having a method to evaluate the likelihood of points close to the submanifold in the observation space by first projecting them onto the manifold.

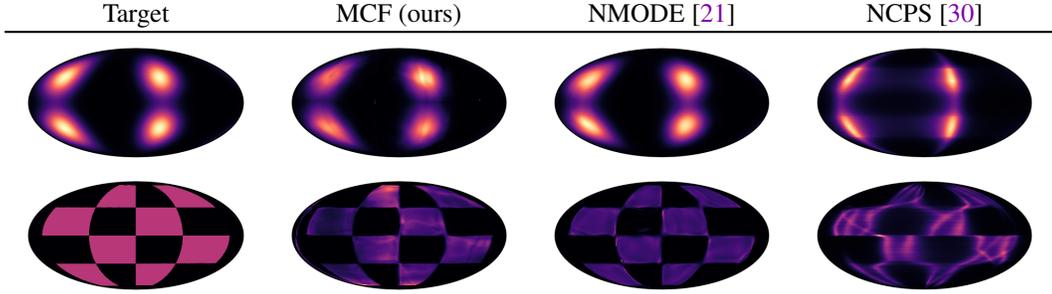


Figure 4: Density estimation on the sphere  $\mathbb{S}^2$  visualized via the Mollweide projection.

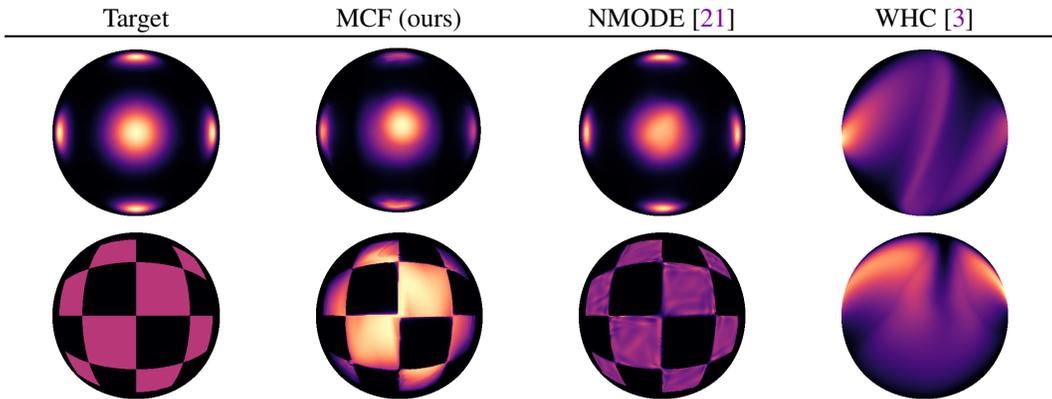


Figure 5: Density estimation on the hyperboloid  $\mathbb{H}^2$  projected on the Poincaré disk.

## 4 Related work

**Learning the manifold structure.** The work closest to ours is that of Brehmer and Cranmer [4]. They, too, split training into two distinct phases. Initially they learn the data manifold via an embedding  $g : \mathcal{M} \rightarrow \mathbb{R}^K$ , which can be considered a composition  $f \circ \phi$  with  $\phi : \mathcal{M} \rightarrow \mathbb{R}^D$  the manifold chart and  $f : \mathbb{R}^D \rightarrow \mathbb{R}^K$  an injective map. Then they learn the density on the manifold via a transformation  $h : \mathbb{R}^D \rightarrow \mathbb{R}^D$ . A crucial limitation is that the data manifold is assumed to be covered by the single chart  $\phi$ , i.e. it is homeomorphic to Euclidean space. The model, thus, cannot represent non-trivial manifolds. Lou et al. [21] proposed another closely related method treating the exponential map as a chart. Since the exponential map  $\exp : T_x \mathcal{M} \rightarrow \mathcal{M}$  is a local diffeomorphism between the (Euclidean) tangent space at  $x$  and the manifold  $\mathcal{M}$ , it is treated as a chart  $\phi$  centered at  $x$ . They learn a vector field in  $T_x \mathcal{M}$  by solving a local ODE for a short time interval  $[t_s, t_e]$ , which is mapped to  $\mathcal{M}$  by the expmap. They use the inverse chart,  $\log : \mathcal{M} \rightarrow T_x \mathcal{M}$ , to map to the new tangent space centered at  $x'$  and repeat the process. In principle, this scheme is general, but in practice, the exponential and logarithmic maps are prohibitively expensive for high dimensional manifolds.

**Flows on fixed manifolds.** A related body of work pertains to flows on manifolds with a priori known topological structure. Rezende et al. [30] construct flows defined on circles, tori and spheres through projective transformations, as well as by adapting Euclidean models, such as autoregressive flows [24] and spline flows [9, 23]. Flow-based models defined in hyperbolic space were presented by Bose et al. [3], wherein two variants are proposed, which use parallel transport of vectors and repeated calls to the exponential and logarithmic maps to map between the tangent bundle and the manifold. A more general method of learning a flow on a manifold was proposed by Gemici et al. [13], which assumes knowledge of a coordinate chart  $\phi : \mathcal{M} \rightarrow \mathbb{R}^D$  and an embedding  $g : \mathcal{M} \rightarrow \mathbb{R}^K$  with  $D < K$ . A limitation here is that  $\mathcal{M}$  needs to be homeomorphic to  $\mathbb{R}^D$ , since it is described by a single chart. When  $\phi$  and  $g$  are learned we arrive at the models presented by Brehmer and Cranmer [4]. We generalize this setting by learning transformations between patches of the manifold  $\mathcal{M}$  and subsets of Euclidean space  $\mathbb{R}^D$ .

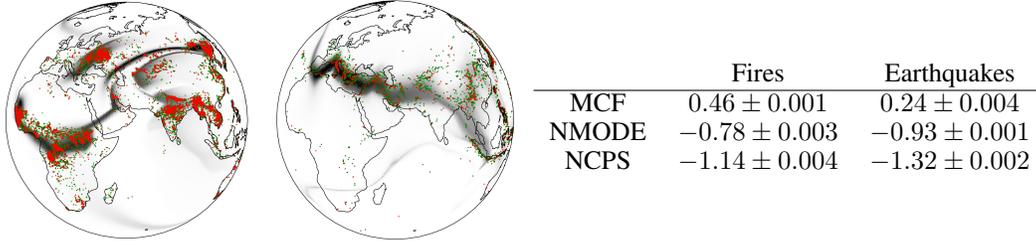


Figure 6: Density estimation results on real world densities. Left: the learned density of the *fires* and *earthquakes* geological data as distributions on a sphere. Green denotes training points, while red denotes evaluation points. Right: kernel density estimation for samples drawn from the model. Higher is better

## 5 Experiments

### 5.1 Qualitative experiments: Estimation of synthetic densities on 2D manifolds

For our first experiment we trained our model, denoted MCF, on synthetic densities on 2D manifolds of well studied topology: the sphere  $\mathbb{S}^2$  and hyperbolic space  $\mathbb{H}^2$ . Our baselines were chosen among models that encode topological information as structural priors by way of a prescribed chart to access the manifold and models that generally rely on the exponential map which still encodes local topological information on the manifold. Of the former we chose the Wrapped Hyperboloid Coupling (denoted by WHC) [3] for  $\mathbb{H}^2$  and the recursive circular spline flow (NCPS) [30] for  $\mathbb{S}^2$ . As for the latter type of models, we chose neural manifold ODEs (NMODE) [21]. Results can be seen in Figures 4 and 5. Our approach achieves improved performance over NCPS and WHC and performs on par with NMODE at significantly reduced running times (see section 5.5). Complete experimental details can be found in Appendix B

### 5.2 Qualitative experiments: Estimation of real world densities on 2D manifolds

We next examine a scenario of real world densities on a low dimensional manifold with known non-Euclidean topology. Our datasets contain the locations of 2 types of natural disasters: earthquakes [12] and fires [1]. These distributions are represented on the sphere  $\mathbb{S}^2$ . They are complex and multimodal and as such, they are suitable for assessing our model’s usefulness in real world scenarios. Figure 6 shows our model’s results. The density learned by our model, while relatively diffuse, captures the modes and general patterns in the data and can serve as a modelling tool which can be subject to further refinement by domain experts. As baselines, we trained NCPS and NMODEs, the same models we trained on spherical densities in section 5.1, but could not achieve satisfactory results. We include them for completeness along with different spherical projections in Appendix C. For a quantitative measure of the learned density, we drew samples from the model and evaluated their density in the ambient space with kernel density estimation, using a Gaussian kernel with bandwidth set to  $\epsilon = 0.1$ . For NCPS we used its evaluated density to construct an importance sampler for points drawn uniformly on the sphere, since the model does not have a generative mode. Results are shown in Figure 6.

### 5.3 Qualitative experiments: Lorenz attractor

Next we model densities residing on topologically complex manifolds. Following Brehmer and Cranmer [4] we illustrate our model’s ability to learn the manifold structure and probability density of the Lorenz system by training on points along sampled trajectories. The stable manifold of the system’s trajectories is a genus 2, two-dimensional manifold embedded in  $\mathbb{R}^3$ . For the classical parameter values, the Lorenz attractor admits a Sinai-Ruelle-Bowen (SRB) measure with support over the surface of the system [33]. Informally, we can say that initial values “diffuse” over this surface.

To create an i.i.d. data set we generated 100 trajectories using the classical parameter values for the system, then uniformly sampled positions  $\mathbf{x}(t) \in \mathbb{R}^3$  for  $t \in [0, 1000]$  along these. The procedure for the creation of the data set matches [4]. Our model takes advantage of rational-quadratic spline

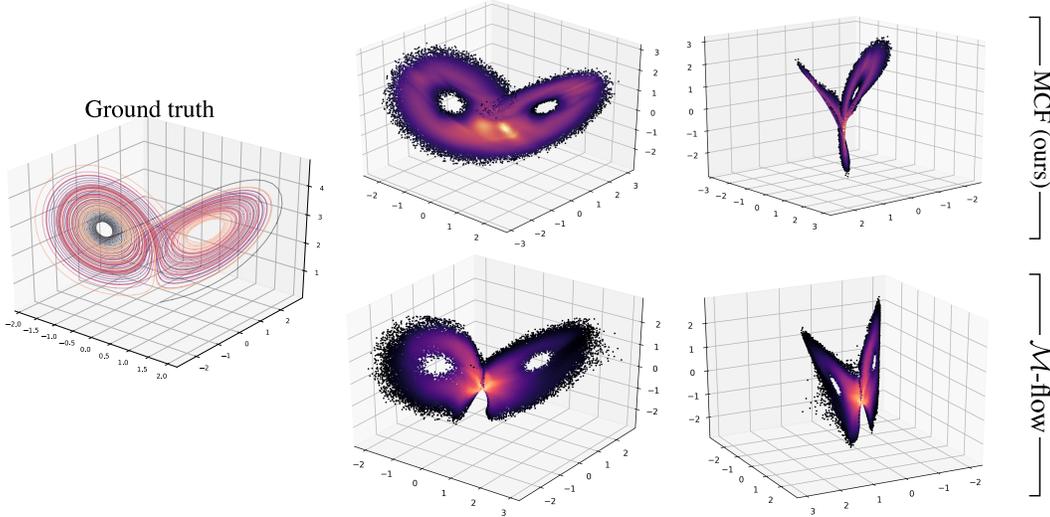


Figure 7: The learned manifold and probability density for the Lorenz attractor system. Brighter color represents areas of higher estimated density. Ground truth shows sampled trajectories from the Lorenz system.

coupling transformations and models the manifold using four coordinate charts. More details on architectures and hyperparameter settings can be found in Appendix D.

Fig. 7 shows the manifold and probability density learned by our model and  $\mathcal{M}$ -flow. Parameterizing the manifold with multiple charts yields visible advantages as our model is able to capture the topology faithfully, without any artifacts, even though the surface is self-intersecting. The single charted  $\mathcal{M}$ -flow struggles to accurately reconstruct the manifold, as it tries to cover the surface with a single coordinate chart, which implies an underlying Euclidean topology.

#### 5.4 Quantitative experiments: Real world particle physics data

Our quantitative experiment focuses on the task of inferring the parameters of a proton-proton collision process at the Large Hadron Collider. Raw data is usually in the order of millions, but following common practice among domain experts, we use a vector of 40 features to represent the data. The model of the process is based on a simulator which generates data  $\mathbf{x} \in \mathbb{R}^{40}$  given parameters  $\theta \in \mathbb{R}^3$  according to an implicit probability distribution  $p(\mathbf{x}|\theta)$ . From domain experts we know that the data resides in a 14-dimensional manifold embedded in  $\mathbb{R}^{40}$ . Given the observations  $\mathbf{x}$  and parameters  $\theta$ , our task is to infer the posterior distribution over the parameters  $p(\theta|\mathbf{x})$ . Thus, we train our model as a conditional density estimator to learn the simulator likelihood function.

Our baselines include a Euclidean flow based on rational quadratic spline transformations [9] (RQ-Flow), the  $\mathcal{M}$ -flow model, as well as an  $\mathcal{M}$ -flow variant with an unrestricted encoder denoted by  $\mathcal{M}_e$ -flow, both of which were proposed by [4]. Furthermore, [4] introduced versions of the models trained with the SCANDAL method [5], which improves inference. All baselines are composed of 35 coupling layers, interspersed with invertible, LU-decomposed linear transformations. The RQ-Flow is trained with maximum likelihood, while the  $\mathcal{M}$ -flow models are trained in two phases, as in [4] corresponding to the manifold learning phase and the density estimation phase. Our model (MCF) is composed of 12 coupling layers, interspersed with invertible, LU-decomposed linear transformations and trained similarly to the  $\mathcal{M}$ -flow variants. We furthermore introduce a variant of MCF, which we fine-tune with early stopping on both phases of training. For further details on architectures and hyperparameters, see Appendix E.

We evaluate our model through a series of metrics. First, we investigate the generative capabilities of all models by evaluating a series of tests on model samples. These ‘‘closure tests’’ are a weighted sum of individual constraints encoding relationships (derived from domain knowledge) between dimensions in the observed vector, taking values in  $[0, 1]$ , where the closer the value is to 0 the higher the quality of the samples. Then, we project test samples on the learned manifold and

Model	sample closure	mean reconstruction error	log posterior
RQ-Flow	<b>0.0019</b> $\pm$ 0.0001	-	-3.94 $\pm$ 0.87
RQ-Flow (SCANDAL)	0.0565 $\pm$ 0.0059	-	-0.49 $\pm$ 0.09
$\mathcal{M}$ -flow	0.0045 $\pm$ 0.0004	<b>0.012</b> $\pm$ 0.001	-1.71 $\pm$ 0.30
$\mathcal{M}$ -flow (SCANDAL)	0.0045 $\pm$ 0.0004	<b>0.011</b> $\pm$ 0.001	0.11 $\pm$ 0.04
$\mathcal{M}_e$ -flow	0.0046 $\pm$ 0.0002	0.029 $\pm$ 0.001	-1.44 $\pm$ 0.34
$\mathcal{M}_e$ -flow (SCANDAL)	0.0291 $\pm$ 0.0010	0.03 $\pm$ 0.002	0.14 $\pm$ 0.09
MCF [ours]	0.031 $\pm$ 0.0024	0.0015 $\pm$ 0.0005	-1.02 $\pm$ 0.14
MCF (fine-tuned) [ours]	0.084 $\pm$ 0.016	<b>0.009</b> $\pm$ 0.002	<b>0.38</b> $\pm$ 0.14

Table 1: Quantitative results on the LHC data. Sample quality is measured by sample closure (lower is better), mean reconstruction error is measured in the ambient/observation space and indicates a model’s manifold learning capability (lower is better), while log-posterior score  $\log p(\theta|\mathbf{x}_{\text{obs}})$  measures quality of inference (higher is better).  $\mathcal{M}$ -flow was proposed by [4], RQ-Flow by [9], while MCF denotes our approach. Baseline results from [4].

measure the reconstruction error, corresponding to the mean squared L2-norm measured in the ambient/observation space. This gives us a measure for the manifold learning capabilities of the models. Finally, we measure the quality of the log posterior inference. Given a set of 20 observed samples  $\mathbf{x}_{\text{obs}} \sim p(\mathbf{x}|\theta^*)$  we use all models to evaluate the likelihood in a Metropolis-Hastings MCMC sampler to generate posterior samples  $\theta \sim p(\theta|\mathbf{x}_{\text{obs}})$ . To evaluate the posterior, we then use kernel density estimation with a Gaussian kernel. We evaluate all models for three different ground truth parameter points  $\theta^*$ . For more details on the experimental setting of the task, see [4].

Table 1 summarizes results for the the Large Hadron Collider data. While the RQ-Flow learns a good sampler for the observed data judging by the closure test score, it does not estimate the density well, as evidenced by the low log posterior estimate. Naive maximum likelihood does not take manifold topology into account, nor does it incentivize models to learn the shape of the embedded data submanifold, rather it relies on models with enough capacity to transform the data in the observation space to a base distribution using invertible maps. To the extent the model manages to learn such a map, this will result in an adequate data sampler but might concurrently lead to overestimating the density of off-the-manifold points. Conversely, models that learn the manifold, project points off-the-manifold onto it, which might yield more accurate density estimates. This is suggested by the results as models that also learn the manifold, result in much better density estimators. The single charted  $\mathcal{M}$ -flow variants achieve better estimates than the RQ-Flow, albeit with worse sample quality. The fine-tuned (early-stopped) variant of our model yields comparable results in the manifold learning task. It results in much better density estimates however, managing to capture a more complex topology than what is suggested by a single chart covering the whole manifold, namely Euclidean topology. Models trained with the SCANDAL scheme invariably trade sample quality off for better density estimates.

## 5.5 Running times

Modelling the topology of the data manifold yields tangible benefits regarding computation time. Since learning the topology of Euclidean patches is easier for a model, our approach generally uses fewer flow layers and parameters than most baselines. Furthermore, with the exception of NCPS and WHC for the synthetic data sets, convergence times are consistently faster than all other baselines even when early stopping is used. Table 2 shows the wallclock times of all models. For the experiments on section 5.1 all models were trained on the CPU since runtimes were significantly faster than on GPUs. For all other experiments all models were trained on a Titan X (Pascal) GPU. Further details can be found in Appendix F.

## 6 Conclusion

We have presented *Multi-chart flows*, a flow-based generative model for modelling data distributions on non-Euclidean manifolds. Recent works in this direction either encode the topology of the target manifold in the model’s architecture, rely on operations that do not scale to moderate and high dimensions or can, in principle, only learn Euclidean manifolds. In contrast, our model can

Datasets	Models		
	MCF (ours)	NMODE	NCPS
Wrapped normals ( $\mathbb{S}^2$ )	$7.21 \pm 0.30$	$54.28 \pm 4.01$	$2.69 \pm 0.42$
Checkerboard ( $\mathbb{S}^2$ )	$3.78 \pm 0.19$	$50.81 \pm 2.74$	$8.13 \pm 0.82$
	MCF (ours)	NMODE	WHC
Five gaussians ( $\mathbb{H}^2$ )	$2.50 \pm 0.05$	$6.48 \pm 2.89$	$0.02 \pm 0.0028$
Checkerboard ( $\mathbb{H}^2$ )	$2.31 \pm 0.17$	$50.88 \pm 9.29$	$0.01 \pm 0.0014$
	MCF (ours)	$\mathcal{M}$ -flow	
Lorenz attractor	$12.05 \pm 0.75$	$35.84 \pm 0.91$	
	MCF (ours)	$\mathcal{M}$ -flow	RQ-Flow
Large Hadron Collider	$12.04 \pm 0.08$	$96.61 \pm 1.93$	$99.74 \pm 4.71$

Table 2: Model wallclock time per dataset (in hours). Computed over 3 training runs with independent initializations.

generalize to non-Euclidean manifolds, learning both their shape and probability densities residing on them. Some of the advantages of our model include faster convergence times and better optima compared to most Euclidean baselines. Shorter convergence times are not surprising since our model does not need a lot of capacity to learn subsets of the data manifold that have simpler (Euclidean) topology and optimization is also easier. Models employing ODEs and making use of local geometric information match and in cases, surpass the performance of our model. This is expected since geometric operations respect manifold topology, and therefore provide a strong inductive bias. Should the bias of these models, however, be incorrect it is detrimental to performance as the bias is served as a hard constraint. Furthermore, these models are inherently at a disadvantage regarding computational cost, since they rely on sequential solvers and do not take full advantage of parallelization. Against models with structural topological priors our model is slower both in wall-clock and convergence time but achieves better optima since it does not rely on classical projection techniques (such as the cylindrical projection as in the case of NCPS [30]) which do not preserve topology.

**Limitations.** Our model relies on rational quadratic (RQ) coupling layers and while we consider the fact that our method is transformation-“agnostic” an advantage, it comes with its own challenges. Specifically, RQ couplings yield unstable log-determinants if left unchecked. In our experiments we have tried various strategies for mitigating this, such as penalizing and clipping gradients, as well as penalizing large log-determinants. All of these methods resulted in similar behaviour and we generally opted for the fastest one. Furthermore, an assumption we make is that the underlying data follows a smooth manifold, which might not necessarily be true. In theory our model would have difficulty modelling a manifold with disconnected components, although in practice it finds a good smooth approximation as evidenced by the checkerboard synthetic densities in section 5.1. Moreover, in the LHC experiments we observed the paradoxical phenomenon where higher sample quality correlates with lower posterior estimates and vice versa. SCANDAL training alleviated this, but it still trades sample quality off for better posterior estimates. This is certainly not new or exclusive to our model but we could not provide a convincing explanation in this work. Finally, quantitative comparisons with other models become harder as different chart parameterizations of manifolds result in different units for the estimated log-likelihood.

**Broader societal impact.** While our model is a more fundamental contribution, it is general enough to be used in more applied scenarios. Specifically, it can be used for learning image manifolds and indeed, we wish to iterate in this direction in future work. While we do not expect our method to immediately perform on par with state-of-the-art methods typically used in the generation of malicious content, it will be part of the broader literature. Given runtime results our method provides a road map for reducing computation times, and by extension, carbon footprints not just for flow models but also deep learning in general. We furthermore showed in our experiments that our approach can provide other disciplines with tools of exploratory and probabilistic analyses for domain specific problems.

## 7 Acknowledgements

We would like to thank Johann Brehmer, Alison Pouplin, Andrea Dittadi and Didrik Nielsen for numerous, insightful conversations.

We are grateful to Johann Brehmer and Kyle Cranmer, and Aaron Lou, Derek Lim, Isay Katsman, Leo Huang and Qingxuang Jiang for making the implementations of [4] and [21] respectively publicly available. We are furthermore grateful to the authors and maintainers of SciPy [17], NumPy [34], scikit-learn [27], PyTorch [26] and matplotlib [15].

This work was supported in part by a research grant (15334) from VILLUM FONDEN. This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement n° 757360).

## References

- [1] EOSDIS (2020). Active fire data. <https://earthdata.nasa.gov/earth-observation-data/near-real-time/firms/active-fire-data>, 2020. Land, Atmosphere Near real-time Capability for EOS (LANCE) system operated by NASA’s Earth Science Data and Information System (ESDIS).
- [2] Wouter Boomsma, Kanti V Mardia, Charles C Taylor, Jesper Ferkinghoff-Borg, Anders Krogh, and Thomas Hamelryck. A generative, probabilistic model of local protein structure. *Proceedings of the National Academy of Sciences*, 105(26):8932–8937, 2008.
- [3] Joey Bose, Ariella Smofsky, Renjie Liao, Prakash Panangaden, and Will Hamilton. Latent variable modelling with hyperbolic normalizing flows. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1045–1055, Virtual, 13–18 Jul 2020. PMLR. URL <http://proceedings.mlr.press/v119/bose20a.html>.
- [4] Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. *arXiv preprint arXiv:2003.13913*, 2020.
- [5] Johann Brehmer, Gilles Louppe, Juan Pavez, and Kyle Cranmer. Mining gold from implicit models to improve likelihood-free inference. *Proceedings of the National Academy of Sciences*, 117(10):5242–5249, 2020.
- [6] Rob Cornish, Anthony Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2133–2143, Virtual, 13–18 Jul 2020. PMLR. URL <http://proceedings.mlr.press/v119/cornish20a.html>.
- [7] Laurent Dinh, Jascha Sohl-Dickstein, Razvan Pascanu, and Hugo Larochelle. A RAD approach to deep mixture models. *CoRR*, abs/1903.07714, 2019. URL <http://arxiv.org/abs/1903.07714>.
- [8] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 3140–3150. Curran Associates, Inc., 2019. URL <https://proceedings.neurips.cc/paper/2019/file/21be9a4bd4f81549a9d1d241981cec3c-Paper.pdf>.
- [9] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 7511–7522. Curran Associates, Inc., 2019. URL <https://proceedings.neurips.cc/paper/2019/file/7ac71d433f282034e088473244df8c02-Paper.pdf>.
- [10] Luca Falorsi and Patrick Forré. Neural ordinary differential equations on manifolds. *arXiv preprint arXiv:2006.06663*, 2020.

- [11] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.
- [12] NOAA National Centers for Environmental Information. Ncei/wds global significant earthquake database. <https://www.ngdc.noaa.gov/hazard/earthqk.shtml>, 2020. National Geophysical Data Center / World Data Service (NGDC/WDS).
- [13] Mevlana C Gemici, Danilo Rezende, and Shakir Mohamed. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*, 2016.
- [14] Thomas Hamelryck, John T Kent, and Anders Krogh. Sampling realistic protein conformations using local structural bias. *PLoS Comput Biol*, 2(9):e131, 2006.
- [15] John D Hunter. Matplotlib: A 2d graphics environment. *IEEE Annals of the History of Computing*, 9(03):90–95, 2007.
- [16] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.
- [17] Eric Jones, Travis Oliphant, Pearu Peterson, et al. Scipy: Open source scientific tools for python. 2001.
- [18] Anuj Karpatne, Imme Ebert-Uphoff, Sai Ravela, Hassan Ali Babaie, and Vipin Kumar. Machine learning for the geosciences: Challenges and opportunities. *IEEE Transactions on Knowledge and Data Engineering*, 31(8):1544–1554, 2018.
- [19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [20] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [21] Aaron Lou, Derek Lim, Isay Katsman, Leo Huang, Qingxuan Jiang, Ser-Nam Lim, and Christopher De Sa. Neural manifold ordinary differential equations. *arXiv preprint arXiv:2006.10254*, 2020.
- [22] Emile Mathieu and Maximilian Nickel. Riemannian continuous normalizing flows. *arXiv preprint arXiv:2006.10605*, 2020.
- [23] Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Trans. Graph.*, 38(5), October 2019. ISSN 0730-0301. doi: 10.1145/3341156. URL <https://doi.org/10.1145/3341156>.
- [24] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017.
- [25] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. pages 1–60, 2019. URL <http://arxiv.org/abs/1912.02762>.
- [26] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.
- [27] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
- [28] David Peel, William J Whiten, and Geoffrey J McLachlan. Fitting mixtures of kent distributions to aid in joint set identification. *Journal of the American Statistical Association*, 96(453):56–63, 2001.

- [29] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR. URL <http://proceedings.mlr.press/v37/rezende15.html>.
- [30] Danilo Jimenez Rezende, George Papamakarios, Sebastien Racaniere, Michael Alberg, Gurtej Kanwar, Phiala Shanahan, and Kyle Cranmer. Normalizing flows on tori and spheres. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8083–8092, Virtual, 13–18 Jul 2020. PMLR. URL <http://proceedings.mlr.press/v119/rezende20a.html>.
- [31] Daniel M Roy, Charles Kemp, Vikash K Mansinghka, and Joshua B Tenenbaum. Learning annotated hierarchies from relational data. In *Advances in neural information processing systems*, pages 1185–1192, 2007.
- [32] Mark Steyvers and Joshua B Tenenbaum. The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive science*, 29(1):41–78, 2005.
- [33] Warwick Tucker. A rigorous ode solver and smale’s 14th problem. *Foundations of Computational Mathematics*, 2(1):53–117, 2002.
- [34] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.

## A Appendix

### A Proof of the lower bound on the data manifold log likelihood

**Proposition 1** *The log likelihood on the data manifold  $\log p(\mathbf{x}) = \log p(\mathbf{u}) - \frac{1}{2} \log \det |G(\mathbf{u})|$  is bounded from below by  $\mathcal{L} = \log p(\mathbf{u}) - \frac{1}{2} \log \text{Tr}(J_{\phi^{-1}}^{\top}(\mathbf{u})J_{\phi^{-1}}(\mathbf{u}))$ .*

**Proof** Denoting the singular values of a matrix by  $s_i$ , we have:

$$\frac{1}{2} \sum_i \log s_i^2 \leq \frac{1}{2} \log \sum_i s_i^2 \quad (11)$$

$$-\frac{1}{2} \sum_i \log s_i^2 \geq -\frac{1}{2} \log \sum_i s_i^2 \quad (12)$$

$$\log p(\mathbf{u}) - \frac{1}{2} \sum_i \log s_i^2 \geq \log p(\mathbf{u}) - \frac{1}{2} \log \sum_i s_i^2 \quad (13)$$

$$\log p(\mathbf{x}) \geq \log p(\mathbf{u}) - \frac{1}{2} \log \sum_i s_i^2 \quad (14)$$

Furthermore for a matrix  $A$ , we have:

$$\text{Tr}(A^{\top}A) = \text{Tr}(U^{\top}\Sigma^{\top}VV^{\top}\Sigma U) = \text{Tr}(\Sigma^{\top}\Sigma UU^{\top}) = \text{Tr}(\Sigma^{\top}\Sigma) = \sum_i s_i^2 \quad (15)$$

with  $U, V$  orthogonal matrices and  $\Sigma$  a diagonal matrix containing the singular values of  $A$ .

Thus, eq. 14 becomes:

$$\log p(\mathbf{x}) \geq \log p(\mathbf{u}) - \frac{1}{2} \log \text{Tr}(J_{\phi^{-1}}^{\top}(\mathbf{u})J_{\phi^{-1}}(\mathbf{u})) = \mathcal{L} \quad (16)$$

$$(17)$$

Using Hutchinson’s estimator we can compute the trace efficiently. With  $J_{\phi^{-1}} \in \mathbb{R}^{D \times d}$  and  $p(\mathbf{v}) = \mathcal{N}(0, I_D)$ :

$$\mathcal{L} \approx \log p(\mathbf{u}) - \frac{1}{2} \log \mathbb{E}_{p(\mathbf{v})} \left[ \mathbf{v}^{\top} J_{\phi^{-1}}^{\top}(\mathbf{u})J_{\phi^{-1}}(\mathbf{u})\mathbf{v} \right] \quad (18)$$

$$= \log p(\mathbf{u}) - \frac{1}{2} \log \mathbb{E}_{p(\mathbf{v})} \left[ \|\mathbf{v}^{\top} J_{\phi^{-1}}\|_2^2 \right] \quad (19)$$

## B Details on synthetic 2D experiments

**Datasets** For all target densities in figures 4 and 5, we generated 50000 points for the train set and 10000 points for the validation set. For details on generating the datasets, please see [21].

**Architectures** Table 3 shows the architecture details for MCF. All flow layers are implemented as rational quadratic coupling flows. In all cases, our base distribution is a standard normal in the latent space  $U$ .

Baseline implementations are provided by Lou et al. [21] in <https://github.com/CUAI/Neural-Manifold-Ordinary-Differential-Equations>.

**Training** To train MCF we split training in a manifold learning and a maximum likelihood phase for all datasets. During the former we train for 150 epochs, while during the latter we train for 500 epochs. We used a learning rate of  $2 \cdot 10^{-4}$  and a batch size of 128. During reconstruction, to stabilize training, we add a regularization term  $a = (\log \det J_{\phi}(\mathbf{x}) + \log \det J_{\phi^{-1}}(\mathbf{u}))^2$  to the reconstruction

Hyperparameters	Datasets	
	Checkerboard ( $\mathbb{S}^2$ )	Four wrapped Normals ( $\mathbb{S}^2$ )
Charts	5	4
Chart flow layers	2	6
Base flow layers	2	6
Chart bins	32	6
Base bins	32	6
Spline range	[-3, 3]	[-6, 6]
Linear transform	No	random permutation
MLP layers (& units)	2 (100)	2 (64)
Activation	ReLU	tanh
	Checkerboard ( $\mathbb{H}^2$ )	Five Gaussians ( $\mathbb{H}^2$ )
Charts	2	2
Chart flow layers	4	2
Base flow layers	2	2
Chart bins	12	6
Base bins	12	32
Spline range	[-4, 4]	[-4, 4]
Linear transform	No	No
MLP layers (& units)	2 (64)	2 (100)
Activation	ReLU	ReLU

Table 3: Architecture details for MCF on all synthetic datasets.

loss, which we multiply by 0.5. During the maximum likelihood phase we clip the gradient norm to 1.

To train NCPS we used a learning rate of  $10^{-3}$  and a batch size of 200. For the *spherical checkerboard* dataset we train for 10000 epochs, while for the *four wrapped normals* dataset we train for 5000 epochs.

Training WHC proved to be very challenging and we used big batch sizes and slower learning rates in an attempt to stabilize training. As such, we used a batch size of 500 and a learning rate of  $3 \cdot 10^{-4}$ . Even so, training still diverged after a number of epochs, so we checkpoint the model and show the best obtained results in the main text.

For NMODE, on *four wrapped normals* we used a batch size of 200 and a learning rate of  $10^{-2}$ , training for 600 epochs. For the *spherical checkerboard* we used a batch size of 200 and a learning rate of  $10^{-2}$ , training for 700 epochs. For *five gaussians* we train for 500 epochs, using a learning rate of  $10^{-3}$  and a batch size of 200. Finally, for *hyperbolic checkerboard* we train for 3000 epochs, using a learning rate of  $10^{-3}$  and a batch size of 200.

All models are trained with the Adam optimizer [19]. In general, we chose baseline hyperparameters such that we can have the fastest possible convergence without sacrificing training stability. Please note however that this is a different training setting to the one used for NMODE and the other baselines by Lou et al. [21], as they generated a random batch of points on the manifold for every iteration, whereas in our case we generate a fixed, modest amount of training points and iterate on those. We think that while this is a much harder training scenario, it’s also a more realistic one.

## C Details on real world 2D experiments

### C.1 Experimental details

**MCF** For both datasets (fires and earthquakes) our model uses 3 coordinate charts to parameterize the manifold. The chart centers are sampled from a standard normal distribution in the latent space  $U$ . The chart models  $\phi$  and base model  $h$  comprise 2 flow layers each. These are implemented as rational quadratic coupling layers. We use no linear transformation between the flow layers but alternate the masking of the vectors passing through the flow as is typically done when using flows on low

dimensional vector data. We use 5 bins for the  $\phi$  and  $h$  maps in the range  $[-4, 4]$ . Each coupling transform is parameterized by an MLP with 2 hidden layers. Each hidden layer consists of 50 ReLU units. The dimensionality of the index variable for CIF is 2. Our base distribution is a standard normal in the latent space  $U$ .

**Baselines** The architectures of both the NMODE and NCPS baselines are the same as in the synthetic datasets case.

**Datasets** The *fires* dataset consists of 66444 data points, while the *earthquakes* dataset consists of 5883 data points at the time of writing. We shuffle both datasets and keep 80% for the train sets and 20% for the validation sets.

**Training on *fires*** To train our model (MCF) we split training in two phases: manifold learning and maximum likelihood. During the former we train the model for 200 epochs, while during the latter we train for 500 epochs. We used the Adam optimizer with a learning rate of  $3 \cdot 10^{-5}$  and a batch size of 200. During maximum likelihood training we clipped the gradient norm to 2.

We train NCPS with the Adam optimizer for 3000 epochs with a learning rate of  $10^{-3}$  and a batch size of 200.

We train NMODE with the Adam optimizer for 600 epochs with a learning rate  $3 \cdot 10^{-4}$  and a batch size of 500.

**Training on *earthquakes*** Once again, we train MCF with split training as before. We use 1000 epochs for the manifold learning phase and 3000 epochs for the maximum likelihood phase. We use the Adam optimizer with a learning rate of  $3 \cdot 10^{-5}$  and a batch size of 200. During maximum likelihood training we clipped the gradient norm to 2.

We train NCPS with the Adam optimizer for 10000 epochs with a learning rate of  $10^{-3}$  and a batch size of 200.

We train NMODE with the Adam optimizer for 10000 epochs with a learning rate  $3 \cdot 10^{-4}$  and a batch size of 500.

For all baselines in both datasets we decay the learning rate every 1/3d of the total epochs with a scaling factor of 0.1. We found that training was difficult for all models. The hardest model to train was NMODE even though results for both baselines are generally unsatisfactory (see figure 8). Given this, in our choice of hyperparameters we attempted to strike a balance between fast convergence and stable gradient updates. Furthermore, we checkpoint the models to retain the best performing parameter configuration according to validation results. Finally, regarding MCF, for both datasets we add a regularization term  $a = (\log \det J_\phi(\mathbf{x}) + \log \det J_{\phi^{-1}}(\mathbf{u}))^2$  to the reconstruction loss, which we multiply by 0.3 to stabilize training.

## D Details on the Lorenz experiment

**MCF** Our model uses 4 coordinate charts to parameterize the manifold. The chart centers are sampled from a standard normal distribution in the latent space  $U$ . The chart models  $\phi$  comprise 3 flow layers, while the base model  $h$  comprises 2 flow layers. These are implemented as rational quadratic coupling layers, interspersed with random feature permutations. We use 5 bins for both the  $\phi$  and  $h$  maps in the range  $[-6, 6]$ . Each coupling transform is parameterized by a residual network with 2 residual blocks and 2 hidden layers per block. Each hidden layer consists of 64 ReLU units. The dimensionality of the index variable for CIF is 2. Our base distribution is a standard normal in the latent space  $U$ .

**$\mathcal{M}$ -flow** For  $\mathcal{M}$ -flow we reproduced the reference architecture given by Brehmer and Cranmer [4]. Both the chart model and the base model comprise 5 rational quadratic coupling layers, interspersed with random feature permutations. We use 5 bins for both maps in the range  $[-3, 3]$ . Each coupling transform is parameterized by a residual network with 2 residual blocks and 2 hidden layers per block. Each hidden layer consists of 100 ReLU units.

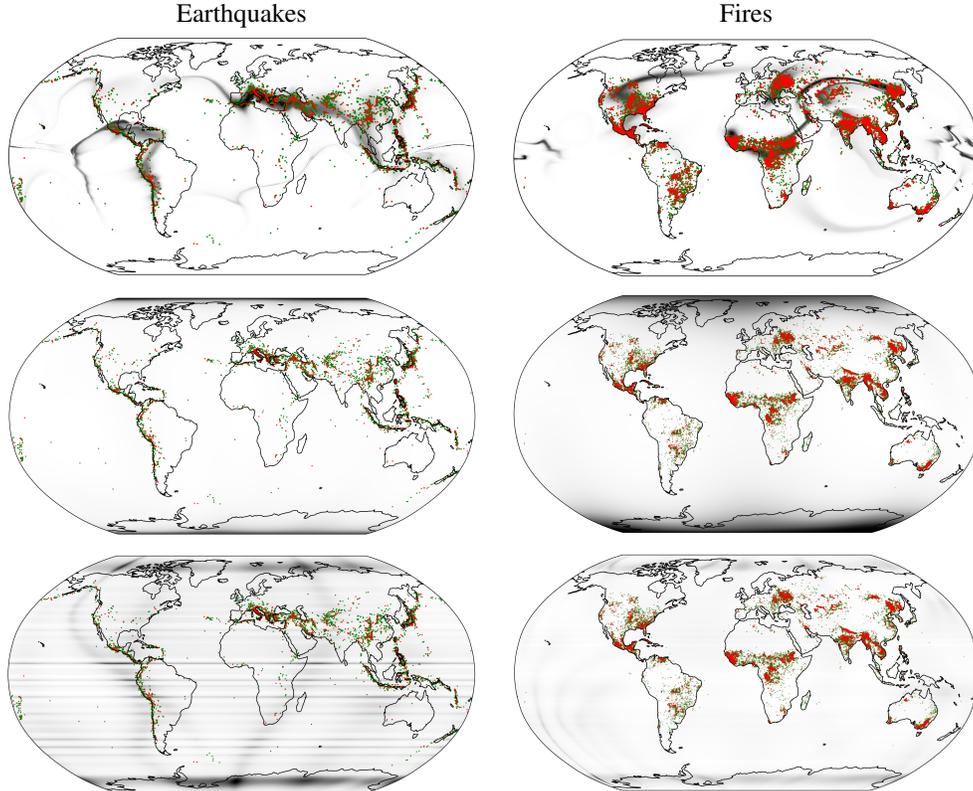


Figure 8: Density estimation results on the earthquakes and fires data. Robinson projection. Top row: *MCF* (ours), middle row: *NMODE*, bottom row: *NCPS*

**Training** Both models were trained on a dataset of  $10^6$  samples with split manifold learning and maximum likelihood training phases. The AdamW optimizer [20] was used in both cases with a learning rate of  $3 \cdot 10^{-4}$ , cosine annealing and weight decay of  $10^{-4}$ . We use a batch size of 100. Initially we used 50 epochs for the manifold learning phase and another 50 epochs for the maximum likelihood phase but we noticed that our model consistently converged in about 1/5th of the available epochs for both phases so ultimately we used only 15 epochs for each training phase for MCF. We checkpoint both models to retain the best performing parameter configuration according to validation results. To stabilize training we add a regularization term  $a = (\log \det J_\phi(\mathbf{x}) + \log \det J_{\phi^{-1}}(\mathbf{u}))^2$  to the reconstruction loss, which we multiply by 0.3. Finally we clip the gradient norm to 2 during the maximum likelihood phase.

## E Details on the Large Hadron Collider experiment

For details on dataset generation, as well as an explanation on the closure tests we refer the interested reader to [4]. The dataset itself can be found in [https://drive.google.com/drive/folders/13x81E08--L8-ORoN\\_QTUbSC\\_frBArPT](https://drive.google.com/drive/folders/13x81E08--L8-ORoN_QTUbSC_frBArPT).

**MCF** Our model uses 4 coordinate charts to parameterize the manifold. The chart centers are sampled from a standard normal distribution in the latent space  $U$ . The chart models  $\phi$  comprise 4 flow layers, while the base model  $h$  comprises 12 layers. These are implemented as rational quadratic coupling layers, interspersed with LU-decomposed invertible linear transformations. We use 11 bins for both maps in the range  $[-10, 10]$ . Each coupling transform is parameterized by a residual network with 2 residual blocks of 2 hidden layers per block. Each hidden layer consists of 100 ReLU units. The dimensionality of the index variable for CIF is 14. Our base distribution is a standard normal in the latent space  $U$

**Baselines** To estimate baseline runtimes we run both RQ-flow and  $\mathcal{M}$ -flow but we note that baseline results are taken from the paper itself. Both baselines are composed of 35 rational quadratic coupling layers, interspersed with LU-decomposed invertible linear transformations. For  $\mathcal{M}$ -flow, the chart model  $\phi$  uses 20 layers and the base model  $h$  uses 15 layers. Each coupling transform is parameterized by a residual network with 2 residual blocks of 2 hidden layers per block. Each hidden layer consists of 100 ReLU units. All runtime estimations are based on the implementation provided by Brehmer and Cranmer [4], which can be found in <https://github.com/johannbrehmer/manifold-flow>.

**Training** We trained our model on the same dataset as [4], using  $10^6$  samples. We used the AdamW optimizer with a learning rate of  $3 \cdot 10^{-4}$ , cosine annealing and a weight decay of  $10^{-5}$ . We split our training in two phases, manifold learning and maximum likelihood. We do not multiply terms in our loss functions with coefficients throughout training as Brehmer and Cranmer [4] do because we found this often destabilized training. In general, we found that our model converges to better optima when trained for fewer iterations, so our fine-tuned version of the model is only trained for 20 epochs with 5 epochs devoted to manifold reconstruction and 15 epochs to density estimation.

**Evaluation** Our evaluation procedure is identical to [4]. In brief, we generate 3 different datasets using 3 different parameter points  $\theta_1 = (0, 0)$ ,  $\theta_2 = (0.5, 0)$  and  $\theta_3 = (-1, -1)$ . Each dataset has 15 i.i.d. samples. For each model and each observed dataset, we generate four MCMC chains of length 750 each, with a Gaussian proposal distribution with mean step size 0.15 and a burn in of 100 steps. Then we obtain kernel density estimates of the log-posterior for each of the 3 parameter points and report the average value in table 1. Like Brehmer and Cranmer [4] we train 5 instances of our model with independent initializations, remove the top and bottom value and report the mean over the remaining runs.

## F Details on Runtimes

Tables 4, 5 & 6 show additional (per epoch) runtime measurements for all models and tasks. NCPS and WHC are faster than our model in per epoch runtimes but converge to worse performing minima. Our construction is generally faster than most baselines and while it is matched in pure per epoch runtimes, it often needs much fewer iterations to converge (as in the case of the Lorenz attractor and the two hyperbolic datasets).

Sphere $\mathbb{S}^2$			
Dataset	Models		
	MCF	NMODE	NCPS
Wrapped normals	$42.20 \pm 4.30$	$225 \pm 0.70$	$3.1 \pm 0.20$
Checkerboard	$20.17 \pm 1.50$	$230 \pm 5.60$	$2.8 \pm 0.60$
Hyperboloid $\mathbb{H}^2$			
Dataset	Models		
	MCF	NMODE	WHC
Five gaussians	$14.62 \pm 0.72$	$15 \pm 1.75$	$7 \pm 0.15$
Checkerboard	$13.86 \pm 2.24$	$20 \pm 2.15$	$8.5 \pm 0.30$

Table 4: Per epoch runtimes in seconds for synthetic datasets. Measured over 500 epochs.

	MCF	$\mathcal{M}$ -flow
Reconstruction phase	$29.09 \pm 1.8$	$30.00 \pm 0.7$
Density phase	$28.7 \pm 2.3$	$29.84 \pm 2.0$

Table 5: Per epoch runtimes in minutes for the Lorenz attractor data, measured over 100 epochs.

	MCF	$\mathcal{M}$ -flow	RQ-Flow
Reconstruction phase	$39.09 \pm 1.8$	$117.8 \pm 4.7$	–
Density phase	$32.7 \pm 2.3$	$110.53 \pm 3.9$	$121.3 \pm 5.5$

Table 6: Per epoch runtimes in minutes for the Large Hadron Collider data, measured over 50 epochs.