Flow-based Generative Models for Learning Manifold to Manifold Mappings

Xingjian Zhen[1], **Rudrasis Chakraborty**[2], **Liu Yang**[1], **Vikas Singh**[1]
[1] University of Wisconsin–Madison
[2] University of California, Berkeley
xzhen3@wisc.edu, rudrasischa@gmail.com, lyang422@wisc.edu, vsingh@biostat.wisc.edu

Abstract

Many measurements or observations in computer vision and machine learning manifest as non-Euclidean data. While recent proposals (like spherical CNN) have extended a number of deep neural network architectures to manifold-valued data, and this has often provided strong improvements in performance, the literature on generative models for manifold data is quite sparse. Partly due to this gap, there are also no modality transfer/translation models for manifold-valued data whereas numerous such methods based on generative models are available for natural images. This paper addresses this gap, motivated by a need in brain imaging – in doing so, we expand the operating range of certain generative models (as well as generative models for modality transfer) from natural images to images with manifold-valued measurements. Our main result is the design of a two-stream version of GLOW (flow-based invertible generative models) that can synthesize information of a field of one type of manifold-valued measurements given another. On the theoretical side, we introduce three kinds of invertible layers for manifold-valued data, which are not only analogous to their functionality in flow-based generative models (e.g., GLOW) but also preserve the key benefits (determinants of the Jacobian are easy to calculate). For experiments, on a large dataset from the Human Connectome Project (HCP), we show promising results where we can reliably and accurately reconstruct brain images of a field of orientation distribution functions (ODF) from diffusion tensor images (DTI), where the latter has a $5\times$ faster acquisition time but at the expense of worse angular resolution.

Introduction

Many measurements in computer vision and machine learning appear in a form that does not satisfy common Euclidean geometry assumptions. Operating on data where the data samples live in structured spaces often leads to situations where even simple operations such as distances, angles and inner products need to be redefined: while occasionally, Euclidean operations may suffice, the error progressively increases depending on the curvature of the space at hand (Feragen, Lauze, and Hauberg 2015). One encounters such data quite often – shapes (Chang et al. 2015), surface normal directions (Straub et al. 2015), graphs and trees (Scarselli et al. 2008; Kipf and Welling 2016) as well as probability distribution functions (Srivastava, Jermyn, and Joshi 2007) are some common examples in vision and computer graphics (Bruno, Conti, and Gregori 2005; Huang et al. 2019). Symmetric positive definite matrices (Moakher 2005; Jayasumana et al. 2013), rotation matrices (Kendall and Cipolla 2017), samples from a sphere (Koppers and Merhof 2016), subspaces/ Grassmannians (Huang, Wu, and Van Gool 2018; Chakraborty, Hauberg, and Vemuri 2017), and a number of other algebraic objects are key ingredients in the design of efficient algorithms in computer vision and medical image analysis as well as in the development or theoretical analysis of various machine learning problems. While a mature literature on extending classical models such as principal components analysis (Dunteman 1989), Kalman filtering (Haykin 2004; Grewal 2011), regression (Fletcher 2013) to such a manifold data regime is available, identifying ways in which deep neural network (DNN) models can be adapted to leverage and utilize the geometry of such data has only become a prominent research topic recently (Bronstein et al. 2017; Chakraborty et al. 2018a; Kondor and Trivedi 2018; Huang, Wu, and Van Gool 2018; Huang and Van Gool 2017). This research direction has already provided convolutional neural networks for various types of manifold measurements (Masci et al. 2015a,b) as well as sequential models such as LSTM (Hochreiter and Schmidhuber 1997)/GRU (Cho et al. 2014) for manifold settings (Jain et al. 2016; Pratiher et al. 2018; Chakraborty et al. 2018b; Zhen et al. 2019).

The results in the literature, so far, on harnessing the power of DNNs for better analysis of manifold or structured data are impressive, but most approaches are discriminative in nature. In other words, the goal is to characterize the conditional distribution $P(Y|\phi(X))$ based on the predictor variables or features X, here X is manifold-valued and the responses or labels Y are Euclidean. The technical thrust is on the design of mechanisms to specify $\phi(\cdot)$ so that it respects the geometry of the data space. In contrast, work on the generative side is very sparse, and to our knowledge, only a couple of methods for a few specific manifolds have been proposed thus far (Brehmer and Cranmer 2020; Rey, Menkovski, and Portegies 2019; Miolane and Holmes 2020; Huang, Wu,

*Code and supplementary materials are available at https://github.com/zhenxingjian/Dual_Manifold_GLOW
An introduction video of this paper is available at https://youtu.be/0r96U0vXsCM

and Van Gool 2019). As a result, the numerous application settings where generative models have shown tremendous promise, namely, semi-supervised learning, data augmentation (Antoniou, Storkey, and Edwards 2017; Radford, Metz, and Chintala 2015) and synthesis of new image samples by modifying a latent variable (Kingma and Dhariwal 2018; Sun et al. 2019) as well as numerous others, currently cannot be easily evaluated for domains with data-types that are not Euclidean vector-valued data.

GANs for Manifold data: what is challenging? There are some reasons why generative models have sparingly been applied to manifold data. A practical consideration is that many application areas where manifold data are common, such as shape analysis and medical imaging, cannot often provide the sample sizes needed to train off-the-shelf generative models such as Generative adversarial networks (GANs) (Goodfellow et al. 2014) and Variational auto-encoders (VAEs) (Kingma and Welling 2013; Doersch 2016). There are also several issues on the technical side. Consider the case where a data sample corresponds to an image where each pixel is a manifold variable (such as a covariance matrix). This means that each sample lives on a product space of the manifold of covariance matrices. In attempting to leverage state of the art methods for GANs such as Wasserstein GANs (WGANs) (Arjovsky, Chintala, and Bottou 2017) will involve, as a first step, defining appropriate generators that take uniformly distributed samples on a product space of manifolds and transforming it into "realistic" samples which are also samples on a product space of manifolds. In principle, this can be attempted via recent developments by extending spherical CNNs or other architectures for manifold data (Chakraborty, Banerjee, and Vemuri 2018). Next, one would not only need to define optimal transport (Fathi and Figalli 2010) or Wasserstein distances (Huang, Wu, and Van Gool 2019) in complicated spaces, but also develop new algorithms to approximate such distances (e.g., Sinkhorn iterations) to make the overall procedure computationally feasible. An interesting attempt to do so was described in (Huang, Wu, and Van Gool 2019). In that paper, Huang et al. introduced a WGAN-based generative model that can generate low-resolution low-dimension manifold-valued images. On the other hand, VAEs are mathematically more convenient in comparison for such data, and as a result, a few recent works show how they can be used for dealing with manifold-valued data (Miolane and Holmes 2020). While these methods inherit VAE's advantages such as ease of synthesis, VAEs are known to suffer from optimization challenges as well as a tendency to generate smoothed samples. It is not clear how the numerical issues, in particular, will be amplified once we move to manifold data where the core operations of calculating geodesics and distances, evaluating derivatives, and so on, must also invoke numerical optimization routines.

Contributions. Instead of GANs or VAEs, the use of flow-based generative models (Rezende and Mohamed 2015; Kingma and Dhariwal 2018), will enable latent variable inference and log-likelihood evaluation. It turns out, as we will show in our development shortly, that the key components (and layers) needed in flow-based generative models with certain mathematical/procedural adjustments, extends nicely to the manifold setting. The goal of this work is to describe our theoretical developments and show promising experiments in brain imaging applications involving manifold-valued data.

Preliminaries

This subsection briefly summarizes some differential geometric concepts/notations we will use. The reader will find a more comprehensive treatment in (Boothby 1986).

Definition 1 (Riemannian manifold and metric) *Let $(\mathcal{M}, g^{\mathcal{M}})$ be an orientable complete Riemannian manifold with a Riemannian metric g, i.e., $\forall x \in \mathcal{M} : g_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbf{R}$ is a bi-linear symmetric positive definite map, where $T_x\mathcal{M}$ is the tangent space of $\mathcal{M}$ at $x \in \mathcal{M}$. Let $d : \mathcal{M} \times \mathcal{M} \to [0, \infty)$ be the distance induced from the Riemannian metric g.*

Definition 2 *Let $p \in \mathcal{M}$, $r > 0$. Define $\mathcal{B}_r(p) = \{q \in \mathcal{M} | d(p, q) < r\}$ to be an open ball at p of radius r.*

Definition 3 (Local injectivity radius (Groisser 2004)) *The local injectivity radius is defined as $r_{inj}(p) = \sup\{r | Exp_p : (\mathcal{B}_r(\mathbf{0}) \subset T_p\mathcal{M}) \to \mathcal{M}\}$ where Exp_p is defined and is a diffeomorphism onto its image at $p \in \mathcal{M}$. The injectivity radius (Manton 2004) of $\mathcal{M}$ is defined as $r_{inj}(\mathcal{M}) = \inf_{p \in \mathcal{M}} \{r_{inj}(p)\}$.*

Within $\mathcal{B}_r(p)$, where $r \leq r_{\text{inj}}(\mathcal{M})$, the mapping $\text{Exp}_p^{-1} : \mathcal{B}_r(p) \to \mathcal{U} \subset T_p\mathcal{M} \subset \mathbf{R}^m$, is called the inverse Exponential/Log map, m is the dimension of $\mathcal{M}$. For each point $p \in \mathcal{M}$, there exists an open ball $\mathcal{U} = \mathcal{B}_r(q)$ for some $q \in \mathcal{M}$ such that $p \in \mathcal{U}$, where $r = r_{\text{inj}}(\mathcal{M})$. Thus, we can cover $\mathcal{M}$ by an indexed (possibly infinite) cover $\left\{\left(\mathcal{B}_r(q), \text{Exp}_q^{-1}\right)\right\}_{q \in \mathcal{I}}$. This set is an example of a *chart* on $\mathcal{M}$; for an example, see (Krauskopf, Osinga, and Galán-Vioque 2007) and also Fig. 1.

For notational simplicity, we will denote a chart covering $p \in \mathcal{M}$ by Φ_p, since in general, we can use an arbitrary chart instead of an inverse Exponential map. Note that the domain for two chart maps may not necessarily be disjoint.

Given a differentiable function $F : \mathcal{M} \to \mathcal{M}$ defined as $x \mapsto \Psi^{-1}\left(\widetilde{F}\left(\Phi(x)\right)\right)$, where Φ and Ψ are the functions in the chart covering x and $F(x)$ respectively and for some differentiable $\widetilde{F} : \mathbf{R}^m \to \mathbf{R}^m$, the Jacobian of F (denoted by $\frac{dy}{\approx dx}$) is defined as:

$$\frac{dy}{\approx dx} := \frac{\partial \Psi \circ \Phi^{-1}}{\partial \Phi(x)} \frac{d\widetilde{F}}{d\widetilde{x}}\Big|_{\Phi(x)} \tag{1}$$

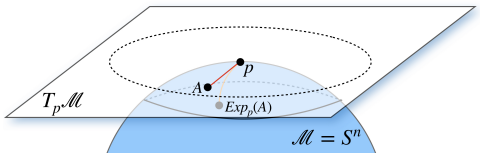

Figure 1: Schematic description of an exemplar manifold ($\mathbf{S}^n$) and the corresponding tangent space at a "pole".

The reason for the peculiar notation is that the derivative cannot be defined on manifold-valued data, so $\frac{dy}{dx}$ is not meaningful: we use the notation $\frac{\cdot}{\approx}$ to acknowledge this difference. Also note that Ψ, Φ are the same only when **(1)** using the global charts for space X and $F(X)$ **(2)** X and $F(X)$ are on the same manifold.

Definition 4 (Group of isometries of $\mathcal{M}$ ($I(\mathcal{M})$)) *A diffeomorphism $\iota : \mathcal{M} \to \mathcal{M}$ is an isometry if it preserves distance, i.e., $d(\iota(x), \iota(y)) = d(x, y)$. The set $I(\mathcal{M})$ forms a group with respect to function composition.*

Rather than writing an isometry as a function ι, we will write it as a group action. Henceforth, let G denote the group $I(\mathcal{M})$, and for $g \in G$, $x \in \mathcal{M}$, let $g \cdot x$ denote the result of applying the isometry g to point x. Similar to the terminologies in (Chakraborty et al. 2018b), we will use the term "translation" to denote the group action ι. This is due to the distance preserving property and is inspired by the analogy from the Euclidean space.

Flow-based Generative Models

In this section, we will introduce flow-based generative models for manifold-valued data. We will first describe the Euclidean formulation and specify which components need to be generalized to get the manifold-valued formulation.

Flow-based Models: Euclidean Case

Flow-based generative models (Rezende and Mohamed 2015; Kingma and Dhariwal 2018; Yang et al. 2019) aim to maximize the log-likelihood of the input data from an unknown distribution. The idea involves mapping the *unknown distribution in the input space* to a *known distribution in the latent space* using an invertible function, f. At a high level, sampling from a known distribution is simpler, so an invertible f can help draw samples from the input space distribution.

Let $\{\mathbf{x}_i\}$ be i.i.d. samples drawn from an unknown distribution $p^*(\mathbf{x})$. Let this unknown distribution be parameterized by θ. In the rest of the paper, we use $p_\theta(\mathbf{x})$ as a proxy for $p^*(\mathbf{x})$. We learn θ over a dataset $\mathcal{D}$. We maximize the likelihood of the model θ given the dataset $\mathcal{D}$ by minimizing the equivalent formulation of negative log-likelihood as:

$$\ell(\theta|\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} -\log p_\theta(\mathbf{x}_i) \tag{2}$$

But to minimize the above expression, we need to know p_θ. One way to bypass this problem is to learn a mapping from a known distribution in the latent space. Let the latent space be $\mathbf{z}$. Then, the generative step is given by $\mathbf{z} \sim p(\mathbf{z})$, $\mathbf{x} = g(\mathbf{z})$. Here $p(\mathbf{z})$ can be a Gaussian distribution $\mathcal{N}(\mathbf{z}; \mathbf{0}, I)$.

Let f be the inverse of g. For normalizing flow (Rezende and Mohamed 2015), f is composed as a sequence of invertible functions $f = f_1 \circ f_2 \circ \ldots \circ f_K$. Hence, we have

$$\mathbf{x} \overset{f_1}{\longleftrightarrow} \mathbf{h}_1 \overset{f_2}{\longleftrightarrow} \mathbf{h}_2 \ldots \overset{f_K}{\longleftrightarrow} \mathbf{z}$$

Using $\mathbf{h}_0 = \mathbf{x}$ and $\mathbf{h}_K = \mathbf{z}$, the log-likelihood of $p_\theta(\mathbf{x})$ is

$$\log p_\theta(\mathbf{x}) = \log p(\mathbf{z}) + \log|\det(d\mathbf{z}/d\mathbf{x})| \tag{3}$$

$$= \log p_\theta(\mathbf{z}) + \sum_{j=0}^{K} \log|\det(d\mathbf{h}_j/d\mathbf{h}_{j-1})| \tag{4}$$

In (Kingma and Dhariwal 2018), the GLOW model is composed of three different layers whose Jacobian $d\mathbf{h}_j/d\mathbf{h}_{j-1}$ is a triangular matrix, simplifying the log-determinant:

$$\log|\det(d\mathbf{h}_j/d\mathbf{h}_{j-1})| = \mathrm{sum}(\log|\mathrm{diag}(d\mathbf{h}_j/d\mathbf{h}_{j-1})|) \tag{5}$$

The three layers in the basic GLOW block (shown in Fig. 2), summarized in Table 1 are all invertible functions. These are (a) *Actnorm*, (b) *Invertible* 1×1 *convolution*, and (c) *Affine Coupling* Layers. Note that the data is squeezed before it is fed into the block. Then, the data is split as in (**?**).

(a) Actnorm normalizes the input to be a zero-mean and identity standard deviation. In (6), μ, σ are initialized from the data and then trained independently.

(b) 1×1 **convolution** applies the invertible matrix R on the channel dimension. In (7), $X \in \mathbf{R}^{s_r \times c_r}$ and $R \in \mathbf{R}^{c_r \times c_r}$ where s_r is the resolution of the input variables while c_r is the number of channels.

(c) Affine Coupling uses the idea of split+concatenation. In (8), the input variable X is *split* along the channel to X_a, X_b, and then Y_a, Y_b are *concatenated* to get the final output Y. Here, S (and T) are real-valued matrices of the same dimension as X_b for element-wise scaling (and translation).

In (Kingma and Dhariwal 2018), authors use a closed form for the inverse of these layers. Notice that calculating the determinant of the Jacobian is simple for all these layers except the affine coupling layer in (8) (Table 1).

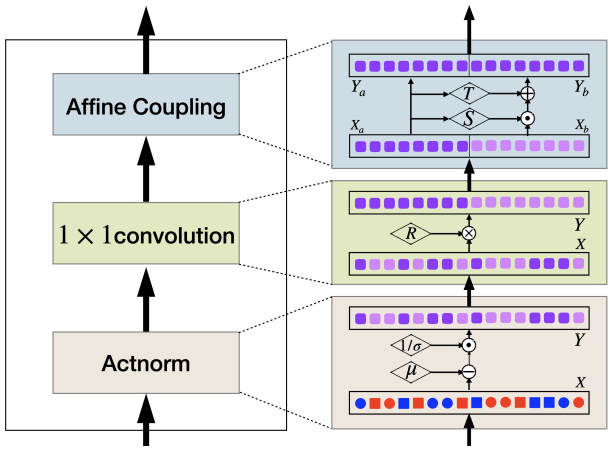

Figure 2: The basic block of GLOW (Kingma and Dhariwal 2018). The color represents the mean while the shape represents the standard deviation. The target distribution on the latent space is the "Grape" rounded rectangles. *Actnorm* normalizes the data to almost "Grape" rounded rectangles, while the disturbance part is "Lavender". 1×1 *convolution* organizes the channels. *Affine Coupling* operates on half of the channels to fit the target distribution. The "$\cdot$" here is the element-wise multiplication, while "$\times$" is the matrix multiplication. "$+/-$" are of the normal definition.

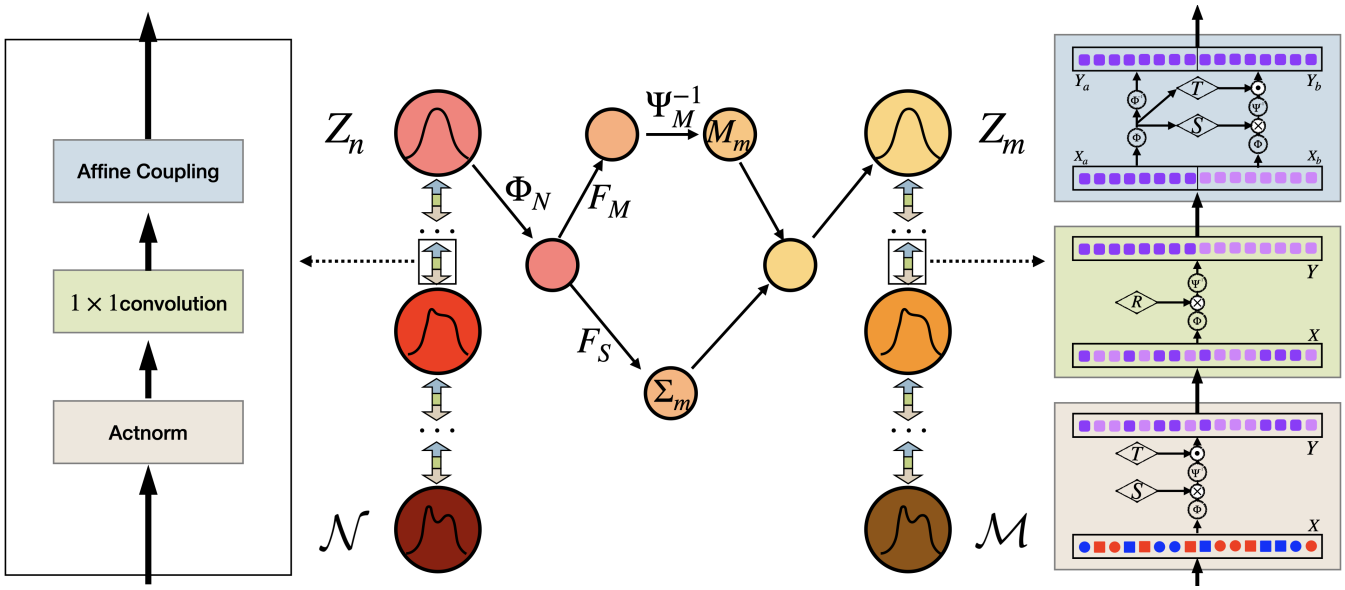

Figure 3: Transfer from the source manifold $\mathcal{N}$ to the target manifold $\mathcal{M}$ with the generative model. The detail of blocks of our model. The meanings of colors and shapes are the same as Fig. 2, while all variables lie on the manifold instead of Euclidean space. The major difference between our manifold-valued GLOW and the original GLOW model is we use a tangent space transformation before and after every operator. Different from Fig. 2, there is no element-wise multiplication. The "·" here is the group operation on the manifold-valued data. The "×" is also the matrix multiplication in the tangent space.

Actnorm	1×1 convolution	Affine Coupling
$Y = \frac{1}{\sigma} \odot (X - \mu)$ (6)	$Y = R \times X$ (7)	$S, T = \mathsf{NN}(X_a)$ $Y_b = S \odot X_b + T$ (8) $Y_a = X_a$

Table 1: Definition of *Actnorm*, 1×1 *convolution* and *Affine Coupling* layers in basic GLOW block. $\odot$ is the elementwise multiplication.The function NN() is a nonlinear mapping.

Since $\frac{dY_a}{dX_b} = 0, \frac{dY_a}{dX_a} = I$, the Jacobian determinant is $\det(S)$.

$$\det\left(\frac{dY}{dX}\right) = \det\left(\begin{bmatrix} \frac{dY_a}{dX_a} & \frac{dY_a}{dX_b} \\ \frac{dY_b}{dX_a} & \frac{dY_b}{dX_b} \end{bmatrix}\right) = \det\left(\begin{bmatrix} I & 0 \\ \frac{dY_b}{dX_a} & S \end{bmatrix}\right) = \det(S) \quad (9)$$

Next Steps: With the description above, we can now list the key operational components in (6)-(9), which we need to modify for our manifold-valued extension.

Key ingredients: In (6) and (8), the operators are **(i)** elementwise multiplication for σ, S and **(ii)** the addition of bias for μ, T. **(iii)** In (7), we require invertible matrices. **(iv)** Finally, to compute the log-likelihood, we need the calculation of derivative in (9). Thus we can verify that the key ingredients to define the model in GLOW are **(i)** elementwise multiplication; **(ii)** addition of bias; **(iii)** invertible matrix; **(iv)** derivative calculation. In theory, if we can modify those components from Euclidean space to manifolds, we will obtain a flow-based generative model on a Riemannian manifold. Observe that **(i)** and **(iii)** are matrix multiplications, which are non-trivial to define on a manifold. In Def. 3, we can use the chart map to map the manifold to a subspace of $\mathbf{R}^m$ where a matrix multiplication can be used. This also provides a way to solve item **(iv)** based on the chart map. In (1), we show how to compute the Jacobian of a differentiable function F from one manifold to another, respecting to the charts of the manifolds. For the item **(ii)**, adding a bias can be viewed as a "translation" in the Euclidean space, while in Def. 4 we define the translation on manifold-valued data using the group action. With these in hand, we are ready to present our proposed manifold version of these layers next.

Flow-based Models: Riemannian Manifold Case

We will now introduce the manifold counterpart of the key operations. See Table 2 for a summary of functions.

(a) Actnorm. Let s_r be the spatial resolution and c_r be the channel size, $X \in \mathcal{M}^{s_r \times c_r}$. We modify (6) to manifold-valued data using the operators we mentioned above in Key ingredients. The bias term is replaced by the group operators T while the multiplication $1/\sigma$ is replaced by the diagonal matrix S of size $m \times m$ in the space after chart mapping $\Phi(\cdot)$. The layer function is defined as in (10).

Determinant of the Jacobian can be computed as shown below in (16). In general, s_r can be a tuple, i.e., for 3D data, it is a 3 dimensional tuple.

Actnorm	1×1 convolution	Affine Coupling
$Y = \Psi^{-1}\left(S \times \Phi\left(X\right)\right) \cdot T$ (10)	$Y = \Psi^{-1}\left(R \times \Phi\left(X\right)\right)$ (11)	$S, T = \mathsf{NN}\left(\Phi(X_a)\right)$ $Y_b = \Psi^{-1}\left(S \times \Phi\left(X_b\right)\right) \cdot T$ (12) $Y_a = X_a$
$X = \Phi^{-1}\left(S^{-1} \times \Psi\left(Y \cdot T^{-1}\right)\right)$ (13)	$X = \Phi^{-1}\left(R^{-1} \times \Psi\left(Y\right)\right)$ (14)	$S, T = \mathsf{NN}\left(\Phi(Y_a)\right)$ $X_b = \Phi^{-1}\left(S^{-1} \times \Psi\left(Y_b \cdot T^{-1}\right)\right)$ (15) $X_a = Y_a$

Table 2: Definition of *Actnorm*, 1×1 *convolution* and *Affine Coupling* layers in our ManifoldGLOW block, with forward function on the top and reverse function in the bottom. Here Φ and Ψ^{-1} are the Chart Map and its inverse. S is a diagonal matrix, so S^{-1} can be computed elementwise. T^{-1} represents the inverse of the group action. The R is chosen as the rotation matrix. Thus, $R^{-1} = R^T$.

$$\det\left(\frac{dY}{\underset{\approx}{dX}}\right) = \prod_{s_r \times c_r}\left(\prod s_i\right)\det\left(\Psi \circ \Phi^{-1}\right) \tag{16}$$

(b) 1×1 **convolution.** We define a 1×1 convolution to offer the flexibility of interaction between channels. Here R is a $c_r \times c_r$ matrix applied after chart mapping $\Phi(\cdot)$. In general, we can learn any $R \in \mathsf{GL}(c_r)$, i.e., a full rank matrix like in (7). But in practice, maintaining full rank is a hard constraint and may become unbounded. As a regularization, we choose R to be a rotation matrix. This layer function is defined as in (11) using the same notation as in (7).

Determinant of the Jacobian can be computed as shown below in (17). Notice that for R to be a rotation matrix, the contribution from $\det(R)$ is ± 1.

$$\det\left(\frac{dY}{\underset{\approx}{dX}}\right) = \prod_{s_r}\det(R)\det\left(\Psi \circ \Phi^{-1}\right) \tag{17}$$

(c) Affine Coupling. For manifold-valued data, given $X \in \mathcal{M}^{s_r \times c_r}$ (where s_r and c_r are spatial and channel resolutions), we first split the data along the channel dimension, i.e., partition c_r into two parts denoted by $X_a \in \mathcal{M}^{s_r \times c_a}$ and $X_b \in \mathcal{M}^{s_r \times c_b}$, where $c_r = c_a + c_b$. From (8), we need to modify the scaling and translation. Here, $S \in \left(\mathbf{R}^{m \times m}\right)^{s_r \times c_a}$ and $T \in G^{s_r \times c_a}$. These two operators play the same roles as in (8), scaling and translation. We need S to be full rank. If needed, one may use constraints like orthogonality or bounded matrix for numerical stability. After performing the coupling, we simply combine Y_a and Y_b to get $Y \in \mathcal{M}^{s_r \times c_r}$ as our output. This function is defined in (12).

Determinant of the Jacobian can be computed as:

$$\begin{bmatrix} \frac{dY_a}{dX_a} & \frac{dY_a}{dX_b} \\ \underset{\approx}{\frac{dY_b}{dX_a}} & \underset{\approx}{\frac{dY_b}{dX_b}} \end{bmatrix} \tag{18}$$

Similar to (9), observe that $\underset{\approx}{\frac{dY_b}{dX_a}}$ involves taking the gradient of a neural network! But fortunately, we only require the determinant of the Jacobian matrix, and the independence of Y_a on X_b saves the calculation of $\underset{\approx}{\frac{dY_b}{dX_a}}$ since $\underset{\approx}{\frac{dY_a}{dX_b}} = 0$. Thus, given $X, Y \in \mathcal{M}^{s_r \times c_r}$, the Jacobian determinant is given as

$$\det\left(\frac{dY}{\underset{\approx}{dX}}\right) = \prod_{s_r \times c_r}\det\left(S\right)\det\left(\Psi \circ \Phi^{-1}\right) \tag{19}$$

Distribution on the latent space: After the cascaded functional transformations described above, we transform X to the latent space $Z \in \mathcal{M}^{s_h \times c_h}$. We define a Gaussian distribution on Z, namely $P\left(Z; M, \Sigma\right)$, by inducing a multi-variate Gaussian distribution from $\mathbf{R}^m$ as

$$\exp\left(-\frac{\left(\Phi(Z) - \Phi(M)\right)^T \Sigma^{-1}\left(\Phi(Z) - \Phi(M)\right)}{2}\right) \Big/ C(\Sigma) \tag{20}$$

where $M \in \mathcal{M}^{s_h \times c_h}$ and $\Sigma \in \mathsf{SPD}(m)^{s_h \times c_h}$ (SPD denotes a symmetric positive definite matrix). $C(\Sigma)$ is the normalization constant to make the total probability to be 1.

Learning Mappings Between Manifolds

We can now ask the question: *can we draw manifold-valued data conditioned on another manifold-valued sample?* Due to the nature of the invertibility of our generative model, this seems to be possible since all we need to develop, in addition to what has been covered, is a scheme to sample data from Euclidean space conditioned on a vector-valued input.

Recently, extensions of the GLOW model (in a Euclidean setup) have been used to generate samples from space $\mathcal{X}$ conditioned on space $\mathcal{Y}$, see (Sun et al. 2019). In this section, we roughly follow (Sun et al. 2019) by using connections in a latent space but in a manifold setting to generate a sample from a manifold $\mathcal{M}$, conditioned on a sample on manifold $\mathcal{N}$. The underlying assumption is that there exists a (smooth) function from $\mathcal{N}$ to $\mathcal{M}$. The generation steps are as follows.

Step (a): Given variables $X \in \mathcal{M}^{s_x \times c_x}$ and $Y \in \mathcal{N}^{s_y \times c_y}$ with the dimension of the manifolds $\mathcal{M}$ and $\mathcal{N}$ to be m and n respectively, we use the two parallel GLOW models (as discussed above) to get the corresponding latent space. Let it be denoted by Z_m and Z_n respectively.

Step (b): After getting the respective latent spaces, we need to fit a distribution on it. Since we wish to generate samples from $\mathcal{M}$, the distribution on the respective latent space Z_m must be induced from the variables in Z_n, i.e., the latent space for $\mathcal{N}$. We do not have any constraint on the distribution parameters for Z_n, so, we use a Gaussian distribution with a fixed M and Σ on Z_n. The parameters for the Gaussian distribution on Z_m are defined as functions of Z_n. Formally, we define $P\left(Z_m; M_m, \Sigma_m\right)$ using (20), where, $M_m = \Psi_M^{-1}\left(F_M(\Phi_N(Z_n))\right.$ and $\Sigma_m = F_S(\Phi_N(Z_n))$. Here, the two functions F_M and F_S are modeled using a neural network. The scheme is shown in Fig. 3.

Specific examples of manifolds. Finally, in order to implement (10), (11) and (12) mentioned in the previous sections, basic operations specific to a manifold are **(a)** the choice of distance, d, **(b)** the isometry group, G, **(c)** the chart map Φ and its inverse, Φ^{-1}.We use three types of non-Euclidean Riemannian manifolds in the experiments presented in this work (including the supplement section), they are **(a)** hypersphere, $\mathbf{S}^{n-1}$ **(b)** space of positive real numbers, $\mathbf{R}_+$ **(c)** space of $n \times n$ symmetric positive definite matrices ($\mathsf{SPD}(n)$). We give the explicit formulation for the operations in Table 3.

Experiments

We demonstrate the experimental results of our model using two setups. First, we generate texture images based on the local covariances, which serves as a sanity check evaluation relative to another generative model for manifold-valued data available at this time. The second experiment, which is our main scientific focus, generates orientation distribution function (ODF) images (Hess et al. 2006) using diffusion tensor imaging (DTI) (Basser, Mattiello, and LeBihan 1994; Alexander et al. 2007). *Note that, in this setting we construct the DTI scans from under-sampled diffusion directions.*

Baseline. Very recently, the $\mathcal{M}_e$-flow (Brehmer and Cranmer 2020) was introduced, which provides a generative model for manifold-valued data. $\mathcal{M}_e$-flow uses an encoder to encode the manifold-valued data in the high-dimensional space into a low-dimensional Euclidean space. During generation, the model will generate the low-dimensional Euclidean data and warp it back to the manifold in the high-dimensional space. The benefit of this method is that it can learn the dimension of the unknown manifold, including natural images like ImageNet (Deng et al. 2009). But for a known Riemannian manifold, the dimension d of the manifold is fixed. For example, $\mathsf{SPD}(n)$ is of dimension $d = n(n+1)/2$, while $\mathbf{S}^m$ is of dimension $d = m$. Thus, for a known Riemannian manifold, $\mathcal{M}_e$-flow learns the chart using an encoder neural network and applies all the operations in the learned space with (known) dimension d. Another interesting recent proposal, manifoldWGAN, (Huang, Wu, and Van Gool 2019) showed that it is possible to generate 32×32 resolution $\mathsf{SPD}(3)$ matrices using WGAN. Due to the involved calculations needed by WGAN, extending it into high-dimension manifold-valued data including ODF ($\mathbf{S}^m$) will require non-trivial changes. Further, manifoldWGAN in its current form does not deal with conditioning the generated images based on another manifold-valued data but is an interesting future direction to explore.

Now, we present experiments for generating texture images before moving to the more challenging ODF generation task.

	$\mathbf{S}^{n-1}$	$\mathbf{R}_+$	$\mathsf{SPD}(n)$
$d(X,Y)$	$\arccos\left(X^T Y\right)$	$\lvert\log(X/Y)\rvert$	$\lVert\mathsf{logm} X^{-1}Y\rVert$
G	$\mathsf{SO}(n-1)$	$\mathbf{R}\setminus\{0\}$	$\mathsf{SO}(m)$
$\Phi(X)$	$\frac{\theta}{\sin(\theta)}\left(X - P\cos(\theta)\right)$	$\log(X)$	$\mathsf{Chol}(X)$
$\Phi^{-1}(\mathbf{v})$	$P\cos(\lVert\mathbf{v}\rVert) + \frac{\mathbf{v}}{\lVert\mathbf{v}\rVert}\sin(\lVert\mathbf{v}\rVert)$	$\exp(\mathbf{v})$	$\mathbf{v}\mathbf{v}^T$

Table 3: The explicit formulation for the basic operations. Here P is an anchor point for chart map, which can be one of the poles. $\theta = d(P,X)$, and $\mathsf{SO}(m)$ is the group of $m \times m$ special orthogonal matrices. $\mathbf{v} \in \mathbf{R}^{n-1}$. Chol is the Cholesky decomposition.

Generating Texture Images

The earth texture images dataset was introduced in (Yu et al. 2019). The train (and test) set have 896 (and 98) images. All images are augmented by random transformations and cropping to size 64×64. Our goal here is to generate texture images based on the local covariances of the three (R, G, B) channels. So the two manifolds are $\mathsf{SPD}(3)$ (for covariance matrix) and $\mathbf{R}_+^3$ (for texture images). Since $\mathcal{M}_e$-flow can only take the Euclidean data as the "conditioning variable", we vectorize the local covariances as the condition variable for $\mathcal{M}_e$-flow. The dimension of the learned space for $\mathcal{M}_e$-flow is chosen as 64 (default configuration from StyleGAN (Karras et al. 2020)). For our case, we build two parallel manifold-GLOW with 8 blocks on each side. After every 2 blocks, the spatial resolution is reduced to half. In the latent space, we train a residual network with 3 residual blocks to map the distribution of the $\mathsf{SPD}(3)$ to $\mathbf{R}_+^3$. Example results are shown in Fig. 4. Even in this simple setting, due to the encoder in the $\mathcal{M}_e$-flow, the generated images lose sharpness. Our model uses the information of the local covariances to generate superior texture images.

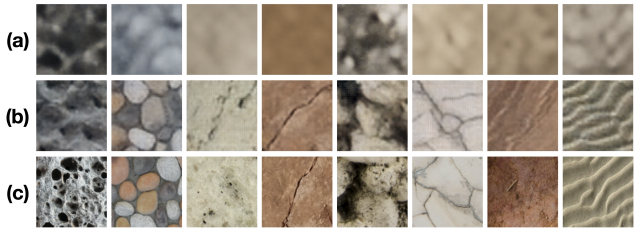

Figure 4: Generated images from (a) $\mathcal{M}_e$-flow, (b) ours, and (c) the ground truth. The condition is the local covariances of the RGB channels.

Main Focus: Diffusion MRI Dataset

Our main focus is the conditional synthesis of structural brain image data. Diffusion-weighted magnetic resonance imaging (dMRI) is an MR imaging modality which measures the diffusion of water molecules at a voxel, and is used to understand brain structural connectivity. Diffusion tensor imaging

(DTI), a type of dMRI (Basser, Mattiello, and LeBihan 1994; Alexander et al. 2007), measures the restricted diffusion of water along only three canonical directions at each voxel. The measurement at each voxel is a symmetric positive definite (SPD) matrix (i.e., manifold-valued data). If multi-shell acquisition capabilities are available, we can obtain a richer acquisition; here, each voxel is an orientation distribution function (ODF) (Hess et al. 2006) which describes the diffusivity in multiple directions (less lossy compared to DTI). By symmetrically/equally sampling 362 points on the continuous distribution function (Garyfallidis et al. 2014), each measurement is a 361-D vector (non-negative entries; sum to 1). Using the square root parameterization (Brody and Hughston 1998; Srivastava, Jermyn, and Joshi 2007), the data at each voxel lies on the positive part of $\mathbf{S}^{361}$ manifold.

We seek to generate a 3D brain image where each voxel is a ODF from the corresponding DTI image (each voxel is a 3×3 SPD matrix). To make the setup more challenging (and scientifically interesting), we generate the DTI images only from randomly under-sampled diffusion directions. We now explain the **(a)** rationale for the application **(b)** data description **(c)** model setup **(d)** evaluations. *Note that in the experiment, since we draw samples from the distribution on the latent space, conditioned on DTI, to get the target representation, we call it generation rather than reconstruction.*

Why generating ODF from DTI is important? For dMRI, different types of acquisitions involve longer/shorter acquisition times. Higher spatial resolution images (e.g., ODF) involves a longer acquisition time (7–12 mins per scan versus 35 mins for an ODF multi-shell scan) and this is problematic, especially for children and the elderly. To shorten the acquisition time with minimal compromise in the image quality, we require mechanisms to transform data acquired from shorter acquisitions (DTI) to a higher spatial resolution image: a field (or image) of ODFs. *This serves as our main motivation.*

However, **(a)** the per voxel degrees of freedom for ODF representation is 361 (lies on $\mathbf{S}^{361}$) while for DTI is 6 (lies on $\mathsf{SPD}(3)$). Hence, it is an ill-posed problem. **(b)** requires mathematical tools to "transform" from one manifold (DTI representation) to another (ODF representation) while preserving structural information. Now, we describe some details of the data, models and present the results.

Dataset. The dataset for our method is the Human Connectome Project (HCP) (Van Essen et al. 2013). The total number of subjects with diffusion measurements available is 1065: 852 were used as training and 213 as the test set. Demographic details are reported in Table 4 (please see (Van Essen et al. 2013) for more details of the dataset). All raw dMRI images are pre-processed with the HCP diffusion pipeline with FSL's 'eddy' (**?**). After correction, ODF and DTI pairs

Dataset	Age				Gender	
	22-25	26-30	31-35	36+	Female	Male
All	224	467	364	10	575(54.0%)	490(46.0%)
Train	178	370	295	9	463(54.3%)	389(45.7%)
Test	46	97	69	1	112(52.6%)	101(47.4%)

Table 4: The demographics used in the study.

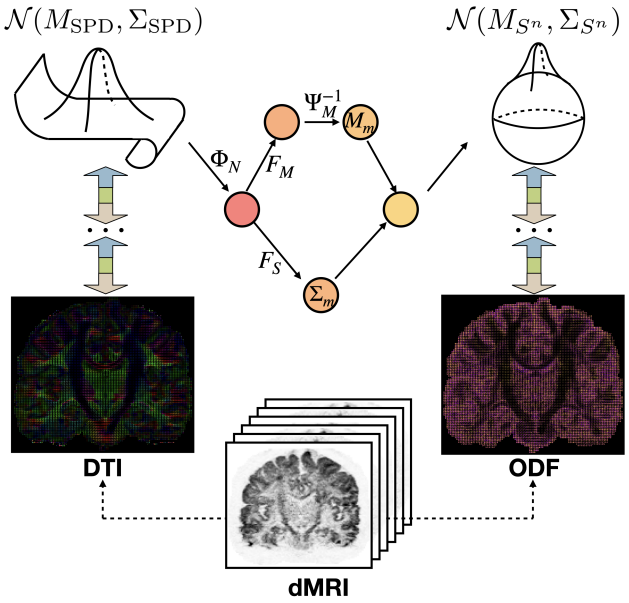

Figure 5: The transformation from DTI to ODF. Both are generated from dMRI. But there might not be dMRI available in some situations. Thus, we want to train the network to transfer DTI to ODF. The latent space is the Gaussian distribution variable.

were obtained using the Diffusion Imaging in Python (DIPY) toolbox (Garyfallidis et al. 2014). Due to the memory requirements of the model and 3D nature of medical data, generation of an ODF image of the entire brain at once remains out of reach at this point, hence we resize the original data into $32 \times 32 \times 32$ but the process can proceed in a sliding window fashion as well.

Reduction in the memory costs. Since the entire 3D models for brain images are still too large to fit into the GPU memory, we need to further simplify the model without sacrificing the performance too much. Recently, NanoFlow (Lee, Kim, and Yoon 2020) was introduced to reduce the number of parameters for sequential data processing. The assumption of NanoFlow is that the Affine Coupling layer, if fully trained, can estimate the distribution for any parts of the input data in a fixed order. There will be some performance drop compared with training different Affine Coupling layers for different parts of the data. But the gain from reducing the parameters is significant. Thus, in our setup, due to the large 3D input, we apply the NanoFlow trick for DTI and ODF separately. For example, for the DTI data, we first split the entire data into 2τ slices called $\{X_1, X_2, ..., X_{2\tau}\}, \tau \geq 1$. Then we can share the two neural networks S and T in the Affine Coupling layer among these slices. The input of two neural networks S and T in Affine Coupling layer would be $X_{2k-1}, k = 1, 2, ..., \tau$, while the output will be the estimated mean and variance of $X_{2k}, k = 1, 2, ..., \tau$ respectively. Due to sharing weights, the number of parameters reduces and becomes feasible for training our 3D DTI and ODF setups.

Model Setup. In order to set up our model, we first build two

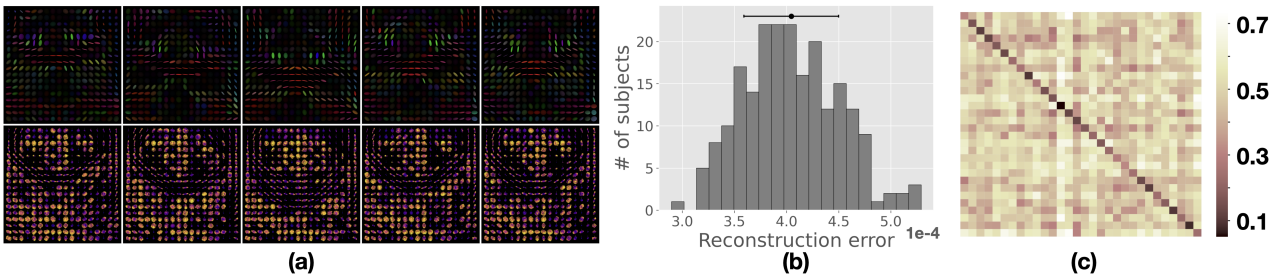

Figure 6: (a) Generated ODF from corresponding DTI. Each pair here contains the input DTI *(top)* and the generated ODF *(bottom)* (b) The distribution of reconstruction error over the testing population. (c) Reconstruction error over the test population shows that the generated image is closest to its own corresponding image (diagonal dominance))

flow-based streams for DTI and ODF separately. Then, in the latent space, we train a transformation operating between the Gaussian distribution variable on the manifold $\mathbf{S}^{361}$ and the Gaussian distribution variable on the manifold $\mathsf{SPD}(3)$. This architecture with two flow-based models and the transformation module can be jointly trained as shown in Fig. 5. We use 6 basic blocks of our manifold GLOW, and after every 2 blocks, reduce the resolution by half. This setup is the same for both DTI and ODF. We use 3 residual network blocks to map the latent space from DTI to ODF. The samples are presented to the model in paired form, i.e., a DTI image (field of SPD matrices) and a corresponding ODF image (a field of ODFs). To reduce the number of parameters for this 3D data, we use a similar idea as NanoFlow (Lee, Kim, and Yoon 2020) that shares the Affine Coupling layer for DTI and ODF separately, with setting $\tau = 32$. As a comparison, for the baseline model $\mathcal{M}_e$-flow, the learned dimension will be $32 \times 32 \times 32 \times d$ where $d = 6$ for DTI and $d = 361$ for ODF. While $\mathcal{M}_e$-flow could be trained for our texture experiments, here, the memory requirements are quite large, quantitatively the number of parameters required for $\mathcal{M}_e$-flow and our model are $1.3e18$ and $2.1e8$ respectively. A similar situation arises in the Euclidean space version of GLOW which also does not leverage the intrinsic Riemmanian metric: therefore, the memory cost will be $6000\times$ more than the natural images which have dimension $224 \times 224 \times 3$. This is infeasible even on clusters and therefore, results from these baselines are very difficult to obtain.

Choice of metrics. We will use "*reconstruction error*" using the distance in Table 3. Although the task here is generation, measuring reconstruction error assesses how "similar" the original ODF is to the generated ODF, generated directly from the corresponding DTI representation. We also perform a group difference analysis to identify statistically different regions across groups (grouped by a dichotomous variable). Since HCP only includes healthy subjects (HCP aging is smaller), we can perform a group difference test based on gender, i.e., *male vs. female*. We evaluate overlap: how/whether group-wise different regions on the generated/reconstructed data agrees with those on the actual ODF images.

Generation results. We present quantitative and qualitative results for generation of ODF from its DTI representation. In Fig. 6(a), we show a few example slices from the given DTI

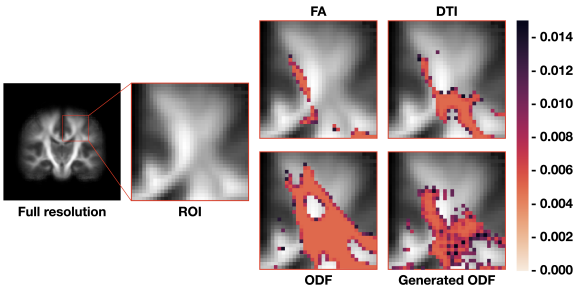

Figure 7: The p-value of one of the ROIs of the entire brain scan with full-resolution. We show that our proposed method can generate meaningful ODF with respect to the group level differences.

and the generated ODF. Overall, the reconstruction error was $4.0(\pm 0.45) \times 1e\text{-}4$. Since perceptually comparing fidelity between generated and ground truth images is difficult, we perform the following quantitative analysis: **(a)** a histogram of the reconstruction error over all 213 test subjects (shown in Fig. 6(b)) **(b)** an error matrix showing how similar the generated ODF image is with the other "incorrect" samples of the population. The goal is to assess if the generated ODF is distinctive across different samples (shown in Fig. 6(c)) . From the histogram presented in Fig. 6(b), we can see that the reconstruction error is consistently low over the entire test population. Now, we generate Fig. 6(c) as follows. For each subject in the test population, we randomly select 29 samples (subjects) from the population and compute the reconstruction error with the generated ODF. This gives us a 30×30 matrix (similar to the confusion matrix). Fig. 6(c) shows the average of 10 runs: lighter shades mean a larger reconstruction error. So, we should ideally see a dark diagonal, which is approximately depicted in the plot. This suggests that for the test population, the generation is meaningful (preserves structures) and distinctive (maintains variability across subjects). There are only few experiments described in the literature on generation of dMRI data (Huang, Wu, and Van Gool 2019; **?**). While (Huang, Wu, and Van Gool 2019) shows the ability to generate 2D (32×32) DTI, the techniques described here can operate on 3D ODF ($\mathbf{S}^{361}$) data and should offer

improvements.

Group difference analysis. We now quantitatively measure if the reconstruction is good enough so that the generated samples can be a good proxy for downstream statistical analysis and yield improvements over the same analysis performed on DTI. We run permutation testing with 10000 independent runs and compute the per-voxel p-value to see which voxels were statistically different between the groups for the following settings **(a)** original ODF **(b)** generated ODF **(c)** DTI **(d)** functional anisotropy (FA) representation (commonly used summary of DTI). Both DTI and FA are commonly used for assessing statistically significant differences across genders (Menzler et al. 2011; Kanaan et al. 2012). But since ODF contains more structural information than either the FA or DTI, our generated ODF should be able to pick up more statistically significant regions over DTI or FA. We evaluate the intersection of significant regions with the original ODF (the original ODF contains the most information). We compute the *intersection over union* (IoU) measure. For the whole brain, FA will have IoU 0.04, while DTI has IoU 0.16. The generated ODF has IoU 0.22. We see that the generated ODF has a larger intersection in the statistically significant regions with the original ODF and offers improvements over DTI. This provides some evidence that the generated ODF preserves the signal that is different across the male/female groups. We also show a zoomed in example of a ROI for the full-resolution images in Fig. 7. The p-values for different ROIs are all < 0.001 in both the original ODF and our generated ODF, indicating consistency of our results, at least in terms of regions identified in downstream statistical analysis. Note that the analysis on the real ODF images serves as the ground truth.

Conclusions

A number of deep neural network formulations have been extended to manifold-valued data in the last two years. While most of these developments are based on models such as CNNs or RNNs, in this work, we study the generative regime: we introduce a flow-based generative model on the Riemannian manifold. We show that the three types of layers, Actnorm, Invertible 1×1 convolution, and Affine Coupling layers in such models, can be generalized/ adapted for manifold-valued data in a way that preserves invertibility. We also show that with the transformation in the latent space between the two manifolds, we can generate manifold-valued data based on the information from another manifold. We demonstrate good generation results in the representation of ODF given DTI on the Human Connectome dataset. While the current formulation shows mathematical feasibility and promising results, additional work on the methodological and the implementation side is needed to reduce the runtime to a level where the tools can be deployed in scientific labs.

Acknowledgements

This research was supported in part by grant 1RF1AG059312-01A1 and NSF CAREER RI #1252725.

References

Alexander, A. L.; Lee, J. E.; Lazar, M.; and Field, A. S. 2007. Diffusion tensor imaging of the brain. *Neurotherapeutics* 4(3): 316–329. 6, 7

Antoniou, A.; Storkey, A.; and Edwards, H. 2017. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340* . 2

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*, 214–223. 2

Basser, P. J.; Mattiello, J.; and LeBihan, D. 1994. MR diffusion tensor spectroscopy and imaging. *Biophysical journal* 66(1): 259–267. 6, 7

Boothby, W. M. 1986. *An introduction to differentiable manifolds and Riemannian geometry*, volume 120. Academic press. 2

Brehmer, J.; and Cranmer, K. 2020. Flows for simultaneous manifold learning and density estimation. *arXiv preprint arXiv:2003.13913* . 1, 6

Brody, D. C.; and Hughston, L. P. 1998. Statistical geometry in quantum mechanics. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454(1977): 2445–2475. 7

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34(4): 18–42. 1

Bruno, R.; Conti, M.; and Gregori, E. 2005. Mesh networks: commodity multihop ad hoc networks. *IEEE communications magazine* 43(3): 123–131. 1

Chakraborty, R.; Banerjee, M.; and Vemuri, B. C. 2018. A CNN for homogneous Riemannian manifolds with applications to Neuroimaging. *arXiv preprint arXiv:1805.05487* . 2

Chakraborty, R.; Bouza, J.; Manton, J.; and Vemuri, B. C. 2018a. Manifoldnet: A deep network framework for manifold-valued data. *arXiv preprint arXiv:1809.06211* . 1

Chakraborty, R.; Hauberg, S.; and Vemuri, B. C. 2017. Intrinsic Grassmann averages for online linear and robust subspace learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6196–6204. 1

Chakraborty, R.; Yang, C.-H.; Zhen, X.; Banerjee, M.; Archer, D.; Vaillancourt, D.; Singh, V.; and Vemuri, B. 2018b. A statistical recurrent model on the manifold of symmetric positive definite matrices. In *Advances in Neural Information Processing Systems*, 8883–8894. 1, 3

Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. 2015. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* . 1

Cho, K.; Van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the properties of neural machine

translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* . 1

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee. 6

Doersch, C. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* . 2

Dunteman, G. H. 1989. *Principal components analysis*. 69. Sage. 1

Fathi, A.; and Figalli, A. 2010. Optimal transportation on non-compact manifolds. *Israel Journal of Mathematics* 175(1): 1–59. 2

Feragen, A.; Lauze, F.; and Hauberg, S. 2015. Geodesic exponential kernels: When curvature and linearity conflict. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3032–3042. 1

Fletcher, P. T. 2013. Geodesic regression and the theory of least squares on Riemannian manifolds. *International journal of computer vision* 105(2): 171–185. 1

Garyfallidis, E.; Brett, M.; Amirbekian, B.; Rokem, A.; Van Der Walt, S.; Descoteaux, M.; and Nimmo-Smith, I. 2014. Dipy, a library for the analysis of diffusion MRI data. *Frontiers in neuroinformatics* 8: 8. 7

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680. 2

Grewal, M. S. 2011. *Kalman filtering*. Springer. 1

Groisser, D. 2004. Newton's method, zeroes of vector fields, and the Riemannian center of mass. *Advances in Applied Mathematics* 33(1): 95–135. 2

Haykin, S. 2004. *Kalman filtering and neural networks*, volume 47. John Wiley & Sons. 1

Hess, C. P.; Mukherjee, P.; Han, E. T.; Xu, D.; and Vigneron, D. B. 2006. Q-ball reconstruction of multimodal fiber orientations using the spherical harmonic basis. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine* 56(1): 104–117. 6, 7

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8): 1735–1780. 1

Huang, R.; Rakotosaona, M.-J.; Achlioptas, P.; Guibas, L.; and Ovsjanikov, M. 2019. OperatorNet: Recovering 3D Shapes From Difference Operators. *arXiv preprint arXiv:1904.10754* . 1

Huang, Z.; and Van Gool, L. 2017. A riemannian network for spd matrix learning. In *Thirty-First AAAI Conference on Artificial Intelligence*. 1

Huang, Z.; Wu, J.; and Van Gool, L. 2018. Building deep networks on Grassmann manifolds. In *Thirty-Second AAAI Conference on Artificial Intelligence*. 1

Huang, Z.; Wu, J.; and Van Gool, L. 2019. Manifold-Valued Image Generation with Wasserstein Generative Adversarial Nets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 3886–3893. 1, 2, 6, 8

Jain, A.; Zamir, A. R.; Savarese, S.; and Saxena, A. 2016. Structural-RNN: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5308–5317. 1

Jayasumana, S.; Hartley, R.; Salzmann, M.; Li, H.; and Harandi, M. 2013. Kernel methods on the Riemannian manifold of symmetric positive definite matrices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 73–80. 1

Kanaan, R. A.; Allin, M.; Picchioni, M.; Barker, G. J.; Daly, E.; Shergill, S. S.; Woolley, J.; and McGuire, P. K. 2012. Gender differences in white matter microstructure. *PloS one* 7(6). 9

Karras, T.; Laine, S.; Aittala, M.; Hellsten, J.; Lehtinen, J.; and Aila, T. 2020. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8110–8119. 6

Kendall, A.; and Cipolla, R. 2017. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5974–5983. 1

Kingma, D. P.; and Dhariwal, P. 2018. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, 10215–10224. 2, 3

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* . 2

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* . 1

Kondor, R.; and Trivedi, S. 2018. On the generalization of equivariance and convolution in neural networks to the action of compact groups. *arXiv preprint arXiv:1802.03690* . 1

Koppers, S.; and Merhof, D. 2016. Direct estimation of fiber orientations using deep learning in diffusion imaging. In *International Workshop on Machine Learning in Medical Imaging*, 53–60. Springer. 1

Krauskopf, B.; Osinga, H. M.; and Galán-Vioque, J. 2007. *Numerical continuation methods for dynamical systems*. Springer. 2

Lee, S.-g.; Kim, S.; and Yoon, S. 2020. NanoFlow: Scalable Normalizing Flows with Sublinear Parameter Complexity. *arXiv preprint arXiv:2006.06280* . 7, 8

Manton, J. H. 2004. A globally convergent numerical algorithm for computing the centre of mass on compact Lie groups. In *ICARCV 2004 8th Control, Automation, Robotics and Vision Conference, 2004.*, volume 3, 2211–2216. IEEE. 2

Masci, J.; Boscaini, D.; Bronstein, M.; and Vandergheynst, P. 2015a. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, 37–45. 1

Masci, J.; Boscaini, D.; Bronstein, M.; and Vandergheynst, P. 2015b. Shapenet: Convolutional neural networks on non-euclidean manifolds. Technical report. 1

Menzler, K.; Belke, M.; Wehrmann, E.; Krakow, K.; Lengler, U.; Jansen, A.; Hamer, H.; Oertel, W. H.; Rosenow, F.; and Knake, S. 2011. Men and women are different: diffusion tensor imaging reveals sexual dimorphism in the microstructure of the thalamus, corpus callosum and cingulum. *Neuroimage* 54(4): 2557–2562. 9

Miolane, N.; and Holmes, S. 2020. Learning Weighted Submanifolds with Variational Autoencoders and Riemannian Variational Autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14503–14511. 1, 2

Moakher, M. 2005. A differential geometric approach to the geometric mean of symmetric positive-definite matrices. *SIAM Journal on Matrix Analysis and Applications* 26(3): 735–747. 1

Pratiher, S.; Chattoraj, S.; Agarwal, S.; and Bhattacharya, S. 2018. Grading Tumor Malignancy via Deep Bidirectional LSTM on Graph Manifold Encoded Histopathological Image. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, 674–681. IEEE. 1

Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* . 2

Rey, L. A. P.; Menkovski, V.; and Portegies, J. W. 2019. Diffusion variational autoencoders. *arXiv preprint arXiv:1901.08991* . 1

Rezende, D. J.; and Mohamed, S. 2015. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770* . 2, 3

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20(1): 61–80. 1

Srivastava, A.; Jermyn, I.; and Joshi, S. 2007. Riemannian analysis of probability density functions with applications in vision. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8. IEEE. 1, 7

Straub, J.; Chang, J.; Freifeld, O.; and Fisher III, J. 2015. A Dirichlet process mixture model for spherical data. In *Artificial Intelligence and Statistics*, 930–938. 1

Sun, H.; Mehta, R.; Zhou, H. H.; Huang, Z.; Johnson, S. C.; Prabhakaran, V.; and Singh, V. 2019. DUAL-GLOW: Conditional Flow-Based Generative Model for Modality Transfer. In *Proceedings of the IEEE International Conference on Computer Vision*, 10611–10620. 2, 5

Van Essen, D. C.; Smith, S. M.; Barch, D. M.; Behrens, T. E.; Yacoub, E.; Ugurbil, K.; Consortium, W.-M. H.; et al. 2013. The WU-Minn human connectome project: an overview. *Neuroimage* 80: 62–79. 7

Yang, G.; Huang, X.; Hao, Z.; Liu, M.-Y.; Belongie, S.; and Hariharan, B. 2019. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, 4541–4550. 3

Yu, N.; Barnes, C.; Shechtman, E.; Amirghodsi, S.; and Lukac, M. 2019. Texture Mixer: A Network for Controllable Synthesis and Interpolation of Texture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 12164–12173. 6

Zhen, X.; Chakraborty, R.; Vogt, N.; Bendlin, B. B.; and Singh, V. 2019. Dilated Convolutional Neural Networks for Sequential Manifold-valued Data. In *Proceedings of the IEEE International Conference on Computer Vision*, 10621–10631. 1