Neural Manifold Ordinary Differential Equations

Aaron Lou [* 1]   Derek Lim [* 1]   Isay Katsman [* 1]   Leo Huang [* 1]   Qingxuan Jiang [1]   Ser-Nam Lim [2]
Christopher De Sa [1]

Abstract

To better conform to data geometry, recent deep generative modelling techniques adapt Euclidean constructions to non-Euclidean spaces. In this paper, we study normalizing flows on manifolds. Previous work has developed flow models for specific cases; however, these advancements hand craft layers on a manifold-by-manifold basis, restricting generality and inducing cumbersome design constraints. We overcome these issues by introducing Neural Manifold Ordinary Differential Equations, a manifold generalization of Neural ODEs, which enables the construction of Manifold Continuous Normalizing Flows (MCNFs). MCNFs require only local geometry (therefore generalizing to arbitrary manifolds) and compute probabilities with continuous change of variables (allowing for a simple and expressive flow construction). We find that leveraging continuous manifold dynamics produces a marked improvement for both density estimation and downstream tasks.

1. Introduction

In a normalizing flow (Rezende & Mohamed, 2015), a complex distribution $p(x)$ is transformed to a simple distribution $\pi(z)$ via a diffeomorphism f (i.e. a differentiable bijective map with a differentiable inverse) with probability values given by the change of variables:

$$\log p(x) = \log \pi(z) - \log \det \left| \frac{\partial f^{-1}}{\partial z} \right|, \qquad z = f(x).$$

To compute this update efficiently, f must be constrained to allow for fast evaluation of the determinant, which in the

absence of additional constraints takes $\mathcal{O}(D^3)$ time (where D is the dimension of z). Furthermore, to efficiently generate samples, f must have a computationally cheap inverse. Existing literature increases the expressiveness of such models under these computational constraints and oftentimes parameterizes f with deep neural networks (Germain et al., 2015; Dinh et al., 2017; Kingma & Dhariwal, 2018; Durkan et al., 2019). An important recent advancement, dubbed the Continuous Normalizing Flow (CNF), constructs f using a Neural Ordinary Differential Equation (ODE) with dynamics g and invokes a continuous change of variables which requires only the trace of the Jacobian of g (Chen et al., 2018; Grathwohl et al., 2019).

Since f is a diffeomorphism, the topologies of the distributions p and π must be equivalent. Furthermore, this topology must conform with the underlying latent space, which previous work mostly assumes to be Euclidean. However, topologically nontrivial data often arise in real world examples such as in quantum field theory (Wirnsberger et al., 2020), motion estimation (Feiten et al., 2013), and protein structure prediction (Hamelryck et al., 2006).

In order to go beyond topologically trivial Euclidean space, one can model the latent space with a *smooth manifold*. An n-dimensional manifold[1] $\mathcal{M}$ can be thought of as an n-dimensional analogue of a surface. Concretely, this is formalized with *charts*, which are smooth bijections $\varphi_x : U_x \to V_x$, where $U_x \subseteq \mathbb{R}^n, x \in V_x \subseteq \mathcal{M}$, that also satisfy a smoothness condition when passing between charts. For charts $\varphi_{x_1}, \varphi_{x_2}$ with corresponding $U_{x_1}, V_{x_1}, U_{x_2}, V_{x_2}$, the composed map $\varphi_{x_2}^{-1} \circ \varphi_{x_1} : \varphi_{x_1}^{-1}(V_{x_1} \cap V_{x_2}) \to \varphi_{x_2}^{-1}(V_{x_1} \cap V_{x_2})$ is a diffeomorphism.

Preexisting manifold normalizing flow works do not generalize to arbitrary manifolds, and many examples present constructions extrinsic to the manifold (see Appendix A). In this work, we solve these issues by introducing Manifold Continuous Normalizing Flows (MCNFs), a manifold analogue of CNFs. Concretely, we:

  (i) introduce Neural Manifold ODEs as a generalization of Neural ODEs.

---

[*]Equal contribution [1]Department of Computer Science, Cornell University, Ithaca, NY, USA [2]Facebook AI, New York, NY, USA. Correspondence to: Aaron Lou <al968@cornell.edu>.

[1]All of our manifolds are assumed to be smooth, so we refer to them simply as manifolds.

(ii) develop a dynamic chart method to realize Neural Manifold ODEs in practice.

(iii) construct Manifold Continuous Normalizing Flows.

2. Neural Manifold Ordinary Differential Equations

A manifold ODE is an equation which relates a curve $\mathbf{z} : [t_s, t_e] \to \mathcal{M}$ to a vector field f and takes the form

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), t) \in T_{\mathbf{z}(t)}\mathcal{M} \qquad \mathbf{z}(t_s) = z_s \quad (1)$$

For the full mathematical background, we point the reader to Appendix B.

To leverage manifold ODEs in a deep learning framework similar to Neural ODEs (Chen et al., 2018), we parameterize the dynamics f by a neural network with parameters θ. We define both forward pass and backward pass gradient computation strategies in this framework. Unlike Neural ODEs, forward and backward computations do not perfectly mirror each other since forward mode computation requires explicit manifold methods, while the backward can be defined solely through an ODE in Euclidean space.

2.1. Forward Mode Integration

The first step in defining our Neural Manifold ODE block is the forward mode integration. We review and select appropriate solvers from the literature; for a more thorough introduction we recommend consulting a text such as Crouch & Grossman (1993); Hairer (2011). Broadly speaking, these solvers can be classified into two groups:

(i) *projection methods* that embed the manifold into Euclidean space $\mathbb{R}^d$, integrate with some base Euclidean solver, and project to the manifold after each step. Projection methods require additional manifold structure; in particular, $\mathcal{M}$ must be the level set of some smooth function $g : \mathbb{R}^d \to \mathbb{R}$.

(ii) *implicit methods* that solve the manifold ODE locally using charts. These methods only require manifold-implicit constructions.

Projection methods are conceptually simple, but ultimately suffer from generality issues. In particular, for manifolds such as the open ball or the upper half-space, there is no principled way to project off-manifold points back on. Furthermore, even in nominally well-defined cases such as the sphere, there may still exist points such as the origin for which the projection is not well-defined.

Implicit methods, by contrast, can be applied to any manifold and do not require a level set representation. Thus,

this approach is more amenable to our generality concerns (especially since we wish to work with, for example, hyperbolic space). However, difficulty in defining charts restricts applicability. Due to this reason, implicit schemes often employ additional structure to define manifold variations of step-based solvers (Bielecki, 2002; Crouch & Grossman, 1993). For example, on a Riemannian manifold one can define a variant of an Euler Method solver with update step $z_{t+\epsilon} = \exp_{z_t}(\epsilon f(z_t, t))$ using the Riemannian exponential map. On a Lie group, there are more advanced Runge-Kutta solvers that use the Lie Exponential map and coordinate frames (Crouch & Grossman, 1993; Hairer, 2011).

2.2. Backward Mode Adjoint Gradient Computation

In order to fully incorporate manifold ODEs into the deep learning framework, we must also efficiently compute gradients. Similar to Chen et al. (2018), we develop an adjoint method to analytically calculate the derivative of a manifold ODE instead of directly differentiating through the solver. Similar to manifold adjoint methods for partial differential equations, we use an ambient space (Zahr et al., 2016).

Theorem 2.1. *Suppose we have some manifold ODE as given in Equation 1 and we define some loss function $L : \mathcal{M} \to \mathbb{R}$. Suppose that there is an embedding of $\mathcal{M}$ in some Euclidean space $\mathbb{R}^d$ and we identify $T_x\mathcal{M}$ with an n-dimensional subspace of $\mathbb{R}^d$. If we define the adjoint state to be $\mathbf{a}(t) = D_{\mathbf{z}(t)}L$, then the adjoint satisfies*

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)D_{\mathbf{z}(t)}f(\mathbf{z}(t), t) \qquad (2)$$

Remark. *This theorem resembles the adjoint method in Chen et al. (2018) precisely because of our ambient space condition. In particular, our curve $\mathbf{z} : [t_s, t_e] \to \mathcal{M}$ can be considered as a curve in the ambient space. Furthermore, we do not lose any generality since such an embedding always exists by the Whitney Embedding Theorem (Whitney, 1935), if we simply set $d = 2n$.*

The full proof of Theorem 2.1 can be found in Appendix C.3. Through this adjoint state, gradients can be derived for other parameters in the equation such as t_s, t_e, initial condition z_s, and weights θ.

3. Dynamic Chart Method

While our above theory is fully expressive and general, in this section we address certain computational issues and augment applicability with our *dynamic chart method*.

The motivation for our dynamic chart method comes from Lezcano-Casado (2019), where the author introduces a *dynamic manifold trivialization* technique for Riemannian gradient descent. Here, instead of applying the traditional Riemannian gradient update $z_{t+1} = \exp_{z_t}(-\eta \nabla_{z_t} f)$ for
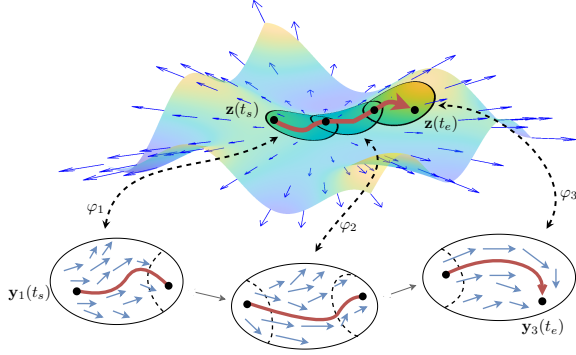
Figure 1. Solving a manifold ODE with our dynamic chart method. We use 3 charts.

Algorithm 1 Dynamic Chart Forward Pass

Given f, local charts φ_x, starting condition z_s and starting/ending times of t_s, t_e.

Initialize $z \leftarrow z_s, \tau \leftarrow t_s$

while $\tau < t_e$ **do**

  Construct an equivalent differential equation $\frac{d\mathbf{y}(t)}{dt} = D_{\varphi_z(\mathbf{y}(t))}\varphi_z^{-1} \circ f(\varphi_z(\mathbf{y}(t)), t)$ with initial condition $\mathbf{y}(\tau) = \varphi_z^{-1}(z)$

  Solve $\mathbf{y}$ locally using some numerical integration technique in Euclidean space. Specifically, solve in some interval $[\tau, \tau + \epsilon]$ for which $\mathbf{z}([\tau, \tau + \epsilon]) \subseteq \text{im}\varphi_z$.

  $z \leftarrow \varphi_z(\mathbf{y}(\tau + \epsilon)), \tau \leftarrow \tau + \epsilon$

end

N time steps, the author repeatedly applies $n \ll N$ local updates. Each update consists of a local diffeomorphism to Euclidean space, n equivalent Euclidean gradient updates, and a map back to the manifold. This allows us to lower the number of expensive exponential map calls and invoke existing Euclidean gradient optimizers such as Adam (Kingma & Ba, 2014). This is similar in spirit to Gemici et al. (2016), but crucially only relies on local diffeomorphisms rather than a global diffeomorphism.

We can adopt this strategy in our Neural Manifold ODE. Specifically, we develop a generalization of the dynamic manifold trivialization for the manifold ODE forward pass. In a similar spirit, we use a local chart to map to Euclidean space, solve an equivalent Euclidean ODE locally, and project back to the manifold using the chart. The full forward pass is given by Algorithm 1 and is visualized Figure 1.

Proofs of correctness and convergence are given in Appendix C.1. Note that this forward integration is implicit, indicating the connections between Hairer & Wanner (2010); Hairer (2011) and Lezcano-Casado (2019). We realize this construction by finding a principled way to pick charts and incorporate this method into neural networks by developing a backward gradient computation.

We can intuitively visualize this dynamic chart forward pass as a sequence of k Neural ODE solvers $\{\text{ODE}_i\}_{i \in [k]}$ with chart transitions $\varphi_{i_2}^{-1} \circ \varphi_{i_1}$ connecting them. Here, we see how the smooth chart transition property comes into play, as passing between charts is the necessary step in constructing a global solution. In addition, this construction provides a *chart-based backpropagation*. Under this dynamic chart forward pass, we can view a Neural Manifold ODE block as the following composition of Neural ODE blocks and chart transitions:

$$\text{MODE} = \varphi_k \circ \text{ODE}_k \circ (\varphi_k^{-1} \circ \varphi_{k-1}) \circ$$
$$\cdots \circ (\varphi_2^{-1} \circ \varphi_1) \circ \text{ODE}_1 \circ \varphi_1^{-1}$$

This allows for gradient computation through backpropagation. We may differentiate through the Neural ODE blocks by the Euclidean adjoint method (Pontryagin et al., 1962; Chen et al., 2018).

To finalize this method, we give a strategy for picking these charts for Riemannian manifolds[2]. As previously mentioned, the exponential map serves as a local diffeomorphism from the tangent space (which can be identified with $\mathbb{R}^n$) and the manifold, so it acts as a chart. Similar to Falorsi et al. (2019), at each point there exists a radius r_x such that $\exp_x$ is a diffeomorphism when restricted to $B_{r_x} := \{v \in T_x\mathcal{M} : \|v\|_\rho < r_x\}$. With this information, we can solve the equivalent ODE with solution $\mathbf{y}$ until we reach a point $y > (1 - \epsilon)\|r_x\|_\rho$, at which point we switch to the exponential map chart defined around $\exp_x(y)$. Complete details are provided in Appendix F.

Our dynamic chart method is a significant advancement over previous implicit methods since we can easily construct charts as we integrate. Furthermore, it provides many practical improvements over vanilla Neural Manifold ODEs. Specifically, we

 (i) *can perform faster evaluations.*

 (ii) *avoid catastrophic gradient instability.*

(iii) *access a wider array of ODE advancements.*

4. Manifold Continuous Normalizing Flows

With our dynamic chart Neural Manifold ODE, we can construct a Manifold Continuous Normalizing Flow (MCNF). The value can be integrated directly through the ODE, so all that needs to be given is the change in log probability. Here, we can invoke continuous change of variables (Chen et al., 2018) on the Neural ODE block and can use the smooth chart transition property (which guarantees that the

---

[2]Note that we do not lose generality since all manifolds can be given a Riemannian metric.

Table 1. MNIST and Omniglot average negative test log likelihood (lower is better) and standard deviation over five trials for varying latent dimension.

		MNIST			Omniglot		
		2	4	6	2	4	6
Euclidean	VAE (Kingma & Welling, 2014)	$143.06 \pm .3$	$117.57 \pm .5$	$102.13 \pm .2$	$154.31 \pm .5$	$143.37 \pm .2$	$138.65 \pm .1$
	RealNVP (Dinh et al., 2017)	$142.09 \pm .7$	$116.32 \pm .7$	$100.95 \pm .1$	$153.93 \pm .5$	$142.98 \pm .3$	$\mathbf{137.21} \pm .1$
	CNF (Grathwohl et al., 2019)	$141.16 \pm .4$	$116.28 \pm .5$	$100.64 \pm .1$	$154.05 \pm .2$	$143.11 \pm .4$	$137.32 \pm .7$
Hyperbolic	HVAE (Mathieu et al., 2019)	$140.04 \pm .9$	$114.81 \pm .8$	$100.45 \pm .2$	$153.97 \pm .3$	$144.10 \pm .8$	$138.02 \pm .3$
	TC (Bose et al., 2020)	$139.58 \pm .4$	$114.16 \pm .6$	$100.45 \pm .2$	157.11 ± 2.7	$143.05 \pm .5$	$137.49 \pm .1$
	WHC (Bose et al., 2020)	140.46 ± 1.3	$113.78 \pm .3$	$100.23 \pm .1$	158.08 ± 1.0	$143.23 \pm .6$	$137.64 \pm .1$
	PRNVP (Gemici et al., 2016)	140.43 ± 1.8	$113.93 \pm .3$	$100.06 \pm .1$	156.71 ± 1.7	$143.00 \pm .3$	$137.57 \pm .1$
	MCNF (ours)	$\mathbf{138.14} \pm .5$	$\mathbf{113.47} \pm .3$	$\mathbf{99.89} \pm .1$	$\mathbf{152.98} \pm .1$	$142.99 \pm .3$	$137.29 \pm .1$

charts are diffeomorphisms) to calculate the final change in probability as:

$$\log p(\text{MODE}) = \log \pi - \sum_{i=1}^{k} \Big[\log \det |D\varphi_i| \tag{3}$$
$$+ \log \det |D\varphi_i^{-1}| + \int \text{tr}(D\varphi_i^{-1} \circ f) \Big]$$

Note that we drop derivative and integration arguments for brevity. A full expression is given in Appendix D.

Since the MCNF requires only the local dynamics (which are in practice parameterized by the exponential map), this means that the construction *generalizes to arbitrary manifolds*. Furthermore, we *avoid diffeomorphism issues*, such as in the case of two antipodal points on the sphere, simply by restricting our chart domains to never include these conjugate points.

5. Experiments

To test our MCNF models, we run density estimation on 2-dimensional hyperbolic and spherical space. For hyperbolic space, the baseline flows are Wrapped Hyperboloid Coupling (WHC) (Bose et al., 2020), and Projected NVP (PRNVP), which learns RealNVP over the projection of the hyperboloid to Euclidean space (Gemici et al., 2016; Dinh et al., 2017). For spherical space, we compare with the recursive flow (NCPS) given in Rezende et al. (2020).

We also test variational inference on hyperbolic space. We train a hyperbolic VAE (Mathieu et al., 2019) and Euclidean VAE (Kingma & Welling, 2014) on Binarized Omniglot (Lake et al., 2015) and Binarized MNIST (Lecun et al., 1998). We compare different hyperbolic and Euclidean flow layers in the respective latent spaces.

Our results are given in Table 1 and Figures 2 and 3. Our MCNF model qualitatively is able to match complex manifold densities significantly better than the baselines. Quantitatively, it achieves better test log likelihood for variational inference.
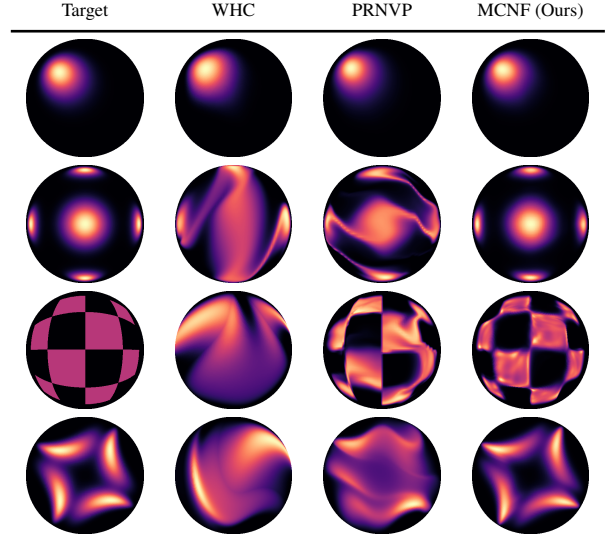

Figure 2. Density estimation on the hyperboloid $\mathbb{H}^2$, which is projected to the Poincaré Ball for visualization.
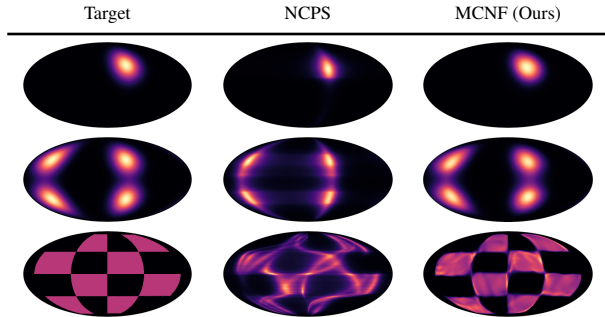

Figure 3. Density estimation on the sphere $\mathbb{S}^2$, which is projected to two dimensions by the Mollweide projection.

6. Conclusion

We have presented Neural Manifold ODEs, which allow for the construction of continuous time manifold neural networks. We hope that our work paves the way for future development of manifold-based deep learning.

7. Acknowledgements

References

Bielecki, A. Estimation of the euler method error on a riemannian manifold. *Communications in Numerical Methods in Engineering*, 18(11):757–763, 2002.

Bose, A. J., Smofsky, A., Liao, R., Panangaden, P., and Hamilton, W. L. Latent variable modelling with hyperbolic normalizing flows. *ArXiv. To appear in ICML 2020*, abs/2002.06336, 2020.

Brehmer, J. and Cranmer, K. Flows for simultaneous manifold learning and density estimation. *ArXiv*, abs/2003.13913, 2020.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. In *NeurIPS*, 2018.

Crouch, P. E. and Grossman, R. Numerical integration of ordinary differential equations on manifolds. *Journal of Nonlinear Science*, 3(1):1–33, 1993. doi: 10.1007/BF02429858. URL https://doi.org/10.1007/BF02429858.

Davidson, T. R., Falorsi, L., Cao, N. D., Kipf, T., and Tomczak, J. M. Hyperspherical variational auto-encoders. In *UAI*, 2018a.

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., and Tomczak, J. M. Hyperspherical variational auto-encoders. *34th Conference on Uncertainty in Artificial Intelligence (UAI-18)*, 2018b.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *ArXiv*, abs/1605.08803, 2017.

do Carmo, M. *Riemannian Geometry*. Mathematics (Boston, Mass.). Birkhäuser, 1992. ISBN 9783764334901. URL https://books.google.com/books?id=uXJQQgAACAAJ.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Neural spline flows. *ArXiv*, abs/1906.04032, 2019.

Falorsi, L., de Haan, P., Davidson, T. R., and Forré, P. Reparameterizing distributions on lie groups. In *AISTATS*, 2019.

Feiten, W., Lang, M., and Hirche, S. Rigid motion estimation using mixtures of projected gaussians. *Proceedings of the 16th International Conference on Information Fusion*, pp. 1465–1472, 2013.

Fisher, N. I., Lewis, T., and Embleton, B. J. J. *Statistical Analysis of Spherical Data*. Cambridge University Press, 1987. doi: 10.1017/CBO9780511623059.

Gemici, M., Rezende, D. J., and Mohamed, S. Normalizing flows on riemannian manifolds. *ArXiv*, abs/1611.02304, 2016.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. Made: Masked autoencoder for distribution estimation. In *ICML*, 2015.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *ArXiv*, abs/1810.01367, 2019.

Hairer, E. Solving ordinary differential equations on manifolds. *Lecture notes, University of Geneva*, 2011.

Hairer, E. and Wanner, G. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2010. ISBN 9783642052217. URL https://books.google.com/books?id=g-jvCAAAQBAJ.

Hamelryck, T., Kent, J. T., and Krogh, A. Sampling realistic protein conformations using local structural bias. *PLoS Computational Biology*, 2, 2006.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. ISSN 0036-8075. doi: 10.1126/science.aab3050. URL https://science.sciencemag.org/content/350/6266/1332.

Lapaine, M. Mollweide map projection. *KoG*, 15(15.):7–16, 2011.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pp. 2278–2324, 1998.

Lee, J. *Riemannian Manifolds: An Introduction to Curvature*. Graduate Texts in Mathematics. Springer New York, 1997. ISBN 9780387982717. URL https://books.google.com/books?id=ZRQgH7FQafgC.

Lee, J. *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics. Springer, 2003. ISBN 9780387954486. URL https://books.google.com/books?id=eqfgZtjQceYC.

Lezcano-Casado, M. Trivializations for gradient-based optimization on manifolds. *arXiv: Learning*, 2019.

Mathieu, E., Lan, C. L., Maddison, C. J., Tomioka, R., and Teh, Y. W. Hierarchical representations with poincaré variational auto-encoders. *ArXiv*, abs/1901.06033, 2019.

Nagano, Y., Yamaguchi, S., Fujita, Y., and Koyama, M. A wrapped normal distribution on hyperbolic space for gradient-based learning. In *ICML*, 2019.

Pontryagin, L., Boltyanskie, V., Collection, K. M. R., Neustadt, L., Trirogoff, K., Gamkrelidze, R., and Mišenko, E. *The Mathematical Theory of Optimal Processes*. Interscience publishers. Interscience Publishers, 1962. URL https://books.google.com/books?id=udNSAAAAMAAJ.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *ArXiv*, abs/1505.05770, 2015.

Rezende, D. J., Papamakarios, G., Racanière, S., Albergo, M. S., Kanwar, G., Shanahan, P. E., and Cranmer, K. Normalizing flows on tori and spheres. *ArXiv. To appear in ICML 2020*, abs/2002.02428, 2020.

Skopek, O., Ganea, O.-E., and Bécigneul, G. Mixed-curvature variational autoencoders. *ArXiv*, abs/1911.08411, 2019.

Stillwell, J. *Naive Lie Theory*. Springer, 2008.

Ulrich, G. Computer generation of distributions on the m-sphere. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 33(2):158–163, 1984. ISSN 00359254, 14679876. URL http://www.jstor.org/stable/2347441.

Wang, P. Z. and Wang, W. Y. Riemannian normalizing flow on variational wasserstein autoencoder for text modeling. *ArXiv*, abs/1904.02399, 2019.

Whitney, H. Differentiable manifolds in euclidean space. *Proceedings of the National Academy of Sciences of the United States of America*, pp. 462–464, 1935.

Wirnsberger, P., Ballard, A. J., Papamakarios, G., Abercrombie, S., Racanière, S., Pritzel, A., Rezende, D. J., and Blundell, C. Targeted free energy estimation via learned mappings. *arXiv: Computational Physics*, 2020.

Zahr, M. J., Persson, P.-O., and Wilkening, J. A fully discrete adjoint method for optimization of flow problems on deforming domains with time-periodicity constraints. *Computers & Fluids*, 139:130–147, 2016.

A. Related Work

In this section we analyze all major preexisting manifold normalizing flows. Previous methods are, in general, hampered by a lack of generality and are burdensomely constructive.

Normalizing Flows on Riemannian Manifolds (Gemici et al., 2016). The first manifold normalizing flow work constructs examples on Riemannian manifolds by first projecting onto Euclidean space, applying a predefined Euclidean normalizing flow, and projecting back. Although simple, this construction is theoretically flawed since the initial manifold projection requires the manifold to be diffeomorphic to Euclidean space. This is not always the case, since, for example, the existence of antipodal points on a sphere necessarily implies that the sphere is not diffeomorphic to Euclidean space[3]. As a result, the construction only works on a relatively small and topologically trivial subset of manifolds.

Our work overcomes this problem by integrating local dynamics to construct a global diffeomorphism. By doing so, we do not have to relate our entire manifold with some Euclidean space, but rather only well-behaved local neighborhoods. We test against Gemici et al. (2016) on hyperbolic space, and our results produce a significant improvement.

Latent Variable Modeling with Hyperbolic Normalizing Flows (Bose et al., 2020). In a recent manifold normalizing flow paper, the authors propose two normalizing flows on hyperbolic space—a specific Riemannian manifold. These models, which they name the Tangent Coupling (TC) and Wrapped Hyperboloid Coupling (WHC), are not affected by the aforementioned problem since hyperbolic space is diffeomorphic to Euclidean space. However, various shortcomings exist. First, in our experiments we find that the methods do not seem to conclusively outperform Gemici et al. (2016). Second, these methods do not generalize to topologically nontrivial manifolds. This means that these flows produce no additional topological complexity and thus the main benefit of manifold normalizing flows is not realized. Third, the WHC construction is not intrinsic to hyperbolic space since it relies on the hyperboloid equation.

Our method, by contrast, is derived from vector fields, which are natural manifold constructs. This not only allows for generality, but also means that our construction respects the manifold geometry. We compare against Bose et al. (2020) on hyperbolic space and find that our results produce a substantial improvement.

Normalizing Flows on Tori and Spheres (Rezende et al., 2020). In another paper, the authors introduce a variety of normalizing flows for tori and spheres. These manifolds are not diffeomorphic to Euclidean space, hence the authors construct explicit global diffeomorphisms. However, their constructions do not generalize and must be intricately designed with manifold topology in mind. In addition, the primary recursive $\mathbb{S}^n$ flow makes use of non-global diffeomorphisms to the cylinder, so densities are not defined everywhere. The secondary exponential map-based $\mathbb{S}^n$ flow is globally defined but is not computationally tractable for higher dimensions.

In comparison, our work is general, requires only local diffeomorphisms, produces globally defined densities, and is tractable for higher dimensions. We test against Rezende et al. (2020) on the sphere and attain markedly better results.

Other Related Work. In Falorsi et al. (2019), the authors define a probability distribution on Lie Groups, a special type of manifold, and as a by-product construct a normalizing flow. The constructed flow is very similar to that found in Gemici et al. (2016), but the authors include a `tanh` nonlinearity at the end of the Euclidean flow to constrain the input space and guarantee that the map back to the manifold is injective. We do not compare directly against this work since Lie Groups are not general (the 2-dimensional sphere is not a Lie Group (Stillwell, 2008)) while Riemannian manifolds are.

There are some related works such as Wirnsberger et al. (2020); Wang & Wang (2019); Brehmer & Cranmer (2020) that are orthogonal to our considerations as they either (i) develop applications as opposed to theory or (ii) utilize normalizing flows as a tool to study Riemannian metrics.

B. Background

In this section, we present background knowledge to establish naming conventions and intuitively illustrate the constructions used for our work. For a more detailed overview, we recommend consulting a text such as Lee (1997; 2003); do Carmo (1992).

---

[3]This example is generalized by the notion of conjugate points. Most manifolds have conjugate points and those without are topologically equivalent to Euclidean space.

B.1. Differential Geometry

Tangent space. For an n-dimensional manifold $\mathcal{M}$, the *tangent space* $T_x\mathcal{M}$ at a point $x \in \mathcal{M}$ is a higher-dimensional analogue of a tangent plane at a point on a surface. It is an n-dimensional real vector space for all points $x \in \mathcal{M}$.

For our purposes, tangent spaces will play two roles. First, they provide a notion of derivation which is crucial in defining manifold ODEs. Second, they will oftentimes be used in place of $\mathbb{R}^n$ for our charts (as we map U_x to some open subset of $T_x\mathcal{M}$ through a change of basis from $\mathbb{R}^n \to T_x\mathcal{M}$).

Pushforward/Differential. A derivative (or a *pushforward*) of a function $f : \mathcal{M} \to \mathcal{N}$ between two manifolds is denoted by $D_x f : T_x\mathcal{M} \to T_x\mathcal{N}$. This is a generalization of the classical Euclidean Jacobian (as $\mathbb{R}^n$ is a manifold), and provides a way to relate tangent spaces at different points.

As one might expect, the pushforward is central in the definition of manifold ODEs (analogous to the importance of the common derivative in Euclidean ODEs). We also use it in our dynamic chart method to map tangent vectors of the manifold to tangent vectors of Euclidean space.

B.2. Riemannian Geometry

While the above theory is general, to concretize some computational aspects (e.g. how to pick charts) and give background on related manifold normalizing flow work, we define relevant concepts from Riemannian geometry.

Riemannian Manifold. The fundamental object of study in Riemannian geometry is the *Riemannian manifold*. This is a manifold with an additional structure called the *Riemannian metric*, which is a smooth metric $\rho_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$ (often denoted as $\langle \cdot, \cdot \rangle_\rho$). This Riemannian metric allows us to construct a distance on the manifold $d_\rho : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$. Furthermore, any manifold can be given a Riemannian metric.

Exponential Map. The *exponential map* $\exp_x : T_x\mathcal{M} \to \mathcal{M}$ can be thought of as taking a vector $v \in T_x\mathcal{M}$ and following the general direction (on the manifold) such that the distance traveled is the length of the tangent vector. Specifically, the distance on the manifold matches the induced tangent space norm $d_\rho(x, \exp_x(v)) = \|v\|_\rho := \sqrt{\langle v, v \rangle_\rho}$. Note that $\exp_x(0) = x$.

The exponential map is crucial in our construction since it acts as a chart. Specifically, if we identify the chart domain with $T_x\mathcal{M}$ then $\exp_x$ is a diffeomorphism when restricted to some local set around 0.

Special Cases. Some special cases of Riemannian manifolds include hyperbolic spaces $\mathbb{H}^n = \{x \in \mathbb{R}^{n+1} : -x_1^2 + \sum_{i=2}^{n+1} x_i^2 = -1, \ x_1 > 0\}$, spheres $\mathbb{S}^n = \{x \in \mathbb{R}^{n+1} : \sum_{i=1}^{n+1} x_i^2 = 1\}$, and tori $\mathbb{T}^n = (\mathbb{S}^1)^n$. Specific formulas for Riemannian computations are given in Appendix D. Hyperbolic space is diffeomorphic to Euclidean space, but spheres and tori are not.

B.3. Manifold Ordinary Differential Equations

Manifold ODE. Finally, we introduce the key objects of study: manifold ordinary differential equations. A manifold ODE is an equation which relates a curve $\mathbf{z} : [t_s, t_e] \to \mathcal{M}$ to a vector field f and takes the form

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), t) \in T_{\mathbf{z}(t)}\mathcal{M} \qquad \mathbf{z}(t_s) = z_s \tag{4}$$

$\mathbf{z}$ is a *solution* to the ODE if it satisfies Equation 4 with initial condition z_s. Similarly to the case of classical ODEs, local solutions are guaranteed to exist under sufficient conditions on f (Hairer, 2011).

C. Proofs of Propositions

C.1. Dynamic Chart Method

Prop C.1 (Correctness). *If $\mathbf{y}(t) : [\tau, \tau + \epsilon] \to \mathbb{R}^n$ is a solution to $\frac{d\mathbf{y}(t)}{dt} = D_{\varphi_z(\mathbf{y}(t))}\varphi_z^{-1} \circ f(\varphi_z(\mathbf{y}(t)), t)$ with initial condition $\mathbf{y}(\tau) = \varphi_z^{-1}(z)$, then $\mathbf{z}(t) = \varphi_z(\mathbf{y}(t))$ is a valid solution to Equation 4 on $[\tau, \tau + \epsilon]$.*

Proof. We see that if $\mathbf{z}(t) = \varphi_z(\mathbf{y}(t))$ then for all $t' \in [\tau, \tau + \epsilon]$

$$\frac{d\mathbf{z}(t')}{dt} = D_{\mathbf{y}(t')}\varphi_z \circ \frac{d\mathbf{y}(t')}{dt} \tag{5}$$

$$= D_{\mathbf{y}(t')}\varphi_z \circ D_{\varphi_z(\mathbf{y}(t'))}\varphi_z^{-1} \circ f(\varphi_z(\mathbf{y}(t')), t') \tag{6}$$

$$= f(\varphi_z(\mathbf{y}(t')), t') \tag{7}$$

$$= f(\mathbf{z}(t'), t') \tag{8}$$

$\square$

Prop C.2 (Convergence). *There exists a finite collection of charts $\{\varphi_i\}_{i=1}^k$ s.t. $\mathbf{z}([t_s, t_e]) \subseteq \bigcup\limits_{i=1}^k \mathrm{im}\varphi_i$.*

Proof. Around each point $z \in \mathbf{z}([t_s, t_e])$ pick a chart φ_z. Then note that $\{\varphi_z\}_{z \in \mathbf{z}([t_s, t_e])}$ satisfies $\mathbf{z}([t_s, t_e]) \subseteq \bigcup\limits_{z} \mathrm{im}\varphi_z$. But, the curve is compact since $[t_s, t_e]$ is compact (and $\mathbf{z}$ is assumed to be continuous) so we can take a finite subset $\{\varphi_i\}_{i=1}^k$ that covers the curve. $\square$

C.2. Derivation of Gradient for Adjoint State

In this section we prove Theorem 2.1. The proof follows from the analogous one in (Chen et al., 2018), though we replace certain operations with their manifold counterparts.

Theorem C.3. *Suppose we have some manifold ODE as given in Equation 4 and we define some loss function $L : \mathcal{M} \to \mathbb{R}$. Suppose that there is an embedding of $\mathcal{M}$ in some Euclidean space $\mathbb{R}^d$ and we identify $T_x\mathcal{M}$ with an n-dimensional subspace of $\mathbb{R}^d$. If we define the adjoint state to be $\mathbf{a}(t) = D_{\mathbf{z}(t)}L$, then the adjoint satisfies*

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)D_{\mathbf{z}(t)}f(\mathbf{z}(t), t) \tag{9}$$

Proof. Consider the first order approximation of $\mathbf{z}(t + \epsilon)$. Since we are embedding $T_x\mathcal{M} \subseteq \mathbb{R}^d$, then under standard $\mathbb{R}^d$ computations we have that

$$\mathbf{z}(t + \epsilon) = \mathbf{z}(t) + \epsilon f(\mathbf{z}(t), t) + \mathcal{O}(\epsilon^2) \tag{10}$$

We set $T_\epsilon(\mathbf{z}(t), t) := \mathbf{z}(t + \epsilon)$. As in the original adjoint method derivation (Chen et al., 2018), we utilize the chain rule

$$D_{\mathbf{z}(t)}L = D_{\mathbf{z}(t+\epsilon)}L \circ D_{\mathbf{z}(t)}T_\epsilon(\mathbf{z}(t), t) \quad \text{or} \quad \mathbf{a}(t) = \mathbf{a}(t + \epsilon)D_{\mathbf{z}(t)}T_\epsilon(\mathbf{z}(t), t) \tag{11}$$

Using these, we get that

$$\frac{d\mathbf{a}(t)}{dt} = \lim_{\epsilon \to 0+} \frac{\mathbf{a}(t + \epsilon) - \mathbf{a}(t)}{\epsilon} \tag{12}$$

$$= \lim_{\epsilon \to 0+} \frac{\mathbf{a}(t + \epsilon) - \mathbf{a}(t + \epsilon)D_{\mathbf{z}(t)}T_\epsilon(\mathbf{z}(t), t)}{\epsilon} \qquad \text{(by Equation 11)} \tag{13}$$

$$= \lim_{\epsilon \to 0+} \frac{\mathbf{a}(t + \epsilon) - \mathbf{a}(t + \epsilon)D_{\mathbf{z}(t)}(\mathbf{z}(t) + \epsilon f(\mathbf{z}(t), t) + \mathcal{O}(\epsilon^2))}{\epsilon} \qquad \text{(by Equation 10)} \tag{14}$$

$$= \lim_{\epsilon \to 0+} \frac{-\epsilon \mathbf{a}(t + \epsilon)D_{\mathbf{z}(t)}f(\mathbf{z}(t), t) + O(\epsilon^2)}{\epsilon} \tag{15}$$

$$= -\mathbf{a}(t)D_{\mathbf{z}(t)}f(\mathbf{z}(t), t) \tag{16}$$

$\square$

D. Computation of Manifold Continuous Normalizing Flows

D.1. Background

In Euclidean space, a normalizing flow f is a diffeomorphism $f : \mathbb{R}^n \to \mathbb{R}^n$ that maps a base probability distribution into a more complex probability distribution. Suppose $z \sim \pi$ is a sample from the simple distribution. By the change of variables formula, the target density of an x in terms of p (the complex distribution) is

$$\log p(x) = \log \pi(z) - \log \det \left| \frac{\partial f^{-1}}{\partial z} \right| \tag{17}$$

There exist a variety of functions f which constrain the log Jacobian determinant to be computationally tractable. An important one is the Continuous Normalizing Flow (CNF). CNFs construct f to be the solution of an ODE (Chen et al., 2018; Grathwohl et al., 2019). Explicitly, let t_0, t_1 be starting and ending times with $t_0 < t_1$, and consider the ordinary differential equation $\frac{d\mathbf{z}(t)}{dt} = g(\mathbf{z}(t), t; \theta)$, where θ parameterizes the dynamics g. For a sample from the base distribution $z \sim \pi$, solving this ODE with initial condition $\mathbf{z}(t_0) = z$ gives the sample from the target distribution $x = \mathbf{z}(t_1)$. The change in the log density given by this model satisfies an ordinary differential equation called the instantaneous change of variables formula:

$$\frac{d \log p(\mathbf{z}(t))}{dt} = -\text{tr}\left(D_{\mathbf{z}(t)} g \right) \tag{18}$$

We can therefore quantify the final probability as

$$\log p(\mathbf{z}(t_1)) = \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \text{tr}\left(D_{\mathbf{z}(t)} g \right) \, dt. \tag{19}$$

D.2. Manifold Continuous Normalizing Flows

For our MCNF we split up the original time $[t_s, t_e]$ into $[t_i, t_{i+1}]$ for $i \in [k]$ where $t_s = t_1 < t_2 < \cdots < t_k < t_{k+1} = t_e$. From our curve $\mathbf{z}$ we can select $z_i = \mathbf{z}(t_i)$, and we have charts $\varphi_i : U_i \to V_i$ s.t. $z_i, z_{i+1} \in V_i$. If our dynamics are determined by $\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), t; \theta)$ then this locally takes the form $\varphi_i(\widehat{f}_i(\varphi_i^{-1}(\mathbf{z}(t), t; \theta)))$, in which $\widehat{f}_i$ is a CNF. The update after passing through a chart φ_i and integrating is given by

$$\log p(z_{i+1}) = \log \pi(z_i) - \left(\log \left| \det D_{\widehat{f}_i(\varphi^{-1}(z_i))} \varphi_i \right| + \int_{t_i}^{t_{i+1}} \text{tr}(D_{\varphi_i^{-1}(z_i)} \widehat{f}_i) dt + \log \left| \det D_{z_i} \varphi_i^{-1} \right| \right)$$

where $\log |\det D\varphi_i|$ is a shorthand for the Riemannian probability update induced by the chart. Note that in general this is not the determinant (for instance when the map goes from elements in $\mathbb{R}^n$ to $\mathbb{R}^d$ where $d > n$). In practice it ends up being quite similar.

We can consider our manifold ODE as a composition of these updates. Therefore, we have that

$$\log p(f(z)) = \\ \log \pi(z) - \sum_{i=1}^{k} \left(\log \left| \det D_{\widehat{f}_i(\varphi^{-1}(z_i))} \varphi_i \right| + \int_{t_i}^{t_{i+1}} \text{tr}(D_{\varphi_i^{-1}(z_i)} \widehat{f}_i) dt + \log \left| \det D_{z_i} \varphi_i^{-1} \right| \right) \tag{20}$$

For our cases we will be setting $\varphi_i = \exp_{z_i}$.

D.3. Base Distributions

Hyperbolic Space. We will use the hyperbolic wrapped normal distribution $\mathcal{G}(\mu, \Sigma)$ where $\mu \in \mathcal{M}$ and $\Sigma \in \mathbb{R}^{n \times n}$ where $\mathcal{M}$ has dimension n (Nagano et al., 2019).
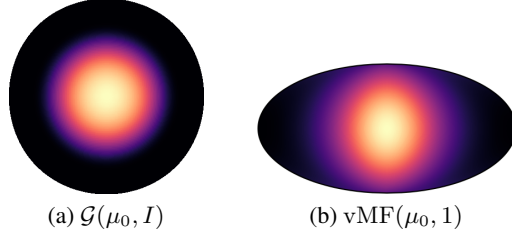
(a) $\mathcal{G}(\mu_0, I)$ (b) $\mathrm{vMF}(\mu_0, 1)$

Figure 4. Base distributions used for flow models. In $\mathbb{H}^2$, $\mu_0 = (1, 0, 0)$ and in $\mathbb{S}^2$, $\mu_0 = (-1, 0, 0)$. (a) is on $\mathbb{H}^2$ as visualized on the Poincaré ball. (b) is on $\mathbb{S}^2$ as visualized by the Mollweide projection.

1. **Sampling.** To sample a $m \in \mathcal{M}$, perform the following steps. A priori, set some $\mu_0 \in \mathcal{M}$. First, sample $v \in \mathcal{N}(\mu_0, \Sigma) \in T_{\mu_0}\mathcal{M}$. Then parallel transport this vector to the mean μ i.e. $u = \mathrm{PT}_{\mu_0 \to \mu}(v)$. Lastly, project to the manifold using the exponential map $m = \exp_\mu(u)$.

2. **Probability Density.** The probability density can be found through the composition of the parallel transport map and exponential map. Specifically, we have that

$$\log p(x) = \log p(v) - \log \left| \det D_u \exp_\mu(u) \right| - \log \left| \det D_v \mathrm{PT}_{\mu_0 \to \mu}(v) \right| \tag{21}$$

Spherical Space. We could possibly perform a relatively similar wrapped normal distribution (Skopek et al., 2019). However, we see that this is theoretically flawed since parallel transport between two conjugate points is not well defined.

Instead, we will use the von Mises-Fisher distribution, a distribution on the $(n-1)$-sphere in $\mathbb{R}^n$ originally derived for use in directional statistics (Fisher et al., 1987). We denote this distribution as $\mathrm{vMF}(\mu, \kappa)$ where $\mu \in \mathbb{R}^n$ is treated as an element of $\mathbb{S}^{n-1}$ via the canonical embedding, and $\kappa \in \mathbb{R}_{\geq 0}$. Note the following about the von Mises-Fisher distribution:

1. **Sampling.** The von Mises-Fisher can be sampled from with efficient methods (Ulrich, 1984; Davidson et al., 2018b).

2. **Probability Density.** From Fisher et al. (1987); Davidson et al. (2018a), we know that the density is given by

$$p(z) = \mathcal{C}_n(\kappa) \exp(\kappa \mu^T z) \tag{22}$$

$$C_n(\kappa) = \frac{\kappa^{n/2-1}}{(2\pi)^{n/2}\mathcal{I}_{n/2-1}(\kappa)} \tag{23}$$

Note that $||\mu||^2 = 1$, $\mathcal{C}_n(\kappa)$ is the normalizing constant, and that $\mathcal{I}_v$ denotes the modified Bessel function of the first kind of order v.

These baseline probability distributions are visualized in Figure 4.

D.4. Hyperbolic Space

D.4.1. ANALYTIC DERIVATIONS

For hyperbolic space, we will work with the hyperboloid $\mathbb{H}^n$. The analytic values of the operations are given in Table 2. Recall that the *Lorentz Inner Product and Norm* are given by

$$\langle x, u \rangle_\mathcal{L} := -x_0 u_0 + x_1 u_1 + \ldots x_n u_n \qquad \|x\|_\mathcal{L} = \sqrt{\langle x, x \rangle_\mathcal{L}} \tag{24}$$

In addition, there are many useful identities which appear in our pipeline.

1. **Stereographic Projection.** To visualize $\mathbb{H}^2$ on the Poincaré ball, we use the stereographic projection as explained in Skopek et al. (2019), which maps a point $(\xi, x^T) \in \mathbb{H}^2 \subseteq \mathbb{R}^3$ to a the point $x/(1 + \xi) \subseteq \mathbb{R}^2$ on the Poincaré ball.

Table 2. Formulas for basic operations in hyperbolic space $\mathbb{H}^n$.

Manifold	$\mathbb{H}^n = \{x \in \mathbb{R}^{n+1} : \|x\|_{\mathcal{L}} = -1, x_0 > 0\}$
Tangent space	$T_x\mathbb{H}^n = \{v \in \mathbb{R}^{n+1} : \langle x, y \rangle_{\mathcal{L}} = 0\}$
Exponential map	$\exp_x(v) = \cosh(\|v\|_{\mathcal{L}})x + \sinh(\|v\|_{\mathcal{L}})\frac{v}{\|v\|_{\mathcal{L}}}$
Logarithmic map	$\log_x(y) = \frac{\text{arccosh}(\langle x,y \rangle_{\mathcal{L}})}{\sinh(\text{arccosh}(\langle x,y \rangle_{\mathcal{L}}))}(y - \langle x, y \rangle_{\mathcal{L}} x)$
Parallel transport	$\text{PT}_{x \to y}(v) = v - \frac{\langle y,v \rangle_{\mathcal{L}}}{1 + \langle x,y \rangle_{\mathcal{L}}}(x + y)$
Tangent projection	$\text{proj}_x(u) = u + \langle x, u \rangle_{\mathcal{L}} x$

Table 3. Formulas for basic operations on the sphere $\mathbb{S}^n$.

Manifold	$\mathbb{S}^n = \{x \in \mathbb{R}^{n+1} : \|x\| = 1\}$
Tangent Space	$T_x\mathbb{S}^n = \{v \in \mathbb{R}^{n+1} : \langle x, v \rangle = 0\}$
Exponential map	$\exp_x(y) = \cos(\|v\|)x + \sin(\|v\|)\frac{v}{\|v\|}$
Logarithmic map	$\log_x(y) = \frac{\arccos(\langle x,y \rangle)}{\sin(\arccos(\langle x,y \rangle))}(y - \langle x, y \rangle x)$
Parallel transport	$\text{PT}_{x \to y}(v) = v - \frac{\langle y,v \rangle}{1 + \langle x,y \rangle}(x + y)$
Tangent projection	$\text{proj}_x(u) = u - \langle x, u \rangle x$

2. **Log Determinants.** The log determinant of the derivative of the exponential map is given by Nagano et al. (2019); Skopek et al. (2019):

$$\log |\det D_v \exp_x| = (n - 1)\left[\log \sinh(\|v\|_{\mathcal{L}}) - \log \|v\|_{\mathcal{L}}\right] \tag{25}$$

The log determinant of the derivative of the log map is the negation of the above by the inverse function theorem, and the log determinant of the derivative of parallel transport is 0.

D.4.2. NUMERICAL STABILITY

In order to ensure numerical stability, we examine several operations which are inherently numerically unstable and present solutions

1. **Arccosh.** Arccosh has a domain of $(-\infty, -1) \cup (1, +\infty)$. In practice, we are concerned with the positive case, although the negative case can be similarly handled. Due to numerical instability a value $1 + \epsilon$ may be realized as 1 in our floating point system. To compensate, we clamp the minimum value to be $1 + \epsilon_0$ for a small fixed ϵ_0.

2. **Sinh Division.** In the exponential and logarithmic maps, there exist terms of the form $\frac{\sinh(x)}{x}$. When $|x| < \epsilon$ for some small ϵ, this is numerically unstable. We special case this (and the derivative) for stability by explicitly deriving the limit value of $x \to 0$ for these cases.

D.5. Spherical Space

D.5.1. ANALYTIC DERIVATIONS

For spherical space, we work with the sphere $\mathbb{S}^n$. The analytic values are given in Table 3. Norms and inner products are assumed to be the Euclidean ℓ^2 values.

Some useful identities are

1. **Mollweide Projection.** To visualize $\mathbb{S}^2$, we use the Mollweide projection that is used in cartography. For latitude θ and longitude φ, the sphere is projected to coordinates (x, y) by (where β is a variable solved for by the given equation) (Lapaine, 2011):

$$2\beta + \sin(2\beta) = \pi \sin(\varphi), \quad x = \frac{2\sqrt{2}}{\pi} \varphi \cos(\beta), \quad y = \sqrt{2} \sin(\beta) \tag{26}$$

2. **Log Determinants.** The log determinant for the exponential map of the Sphere is given by

$$\log |\det D_v \exp_x| = (n-1) \left[\log \sin(\|v\|) - \log \|v\| \right] \tag{27}$$

The log determinant of the derivative of the log map is the negation of the above and the log determinant of the derivative of parallel transport is 0.

D.5.2. NUMERICAL STABILITY

In order to ensure numerical stability, we note that several functions are inherently numerically unstable

1. **Sine division.** In the exp and log maps, there are values of the form $\frac{\sin x}{x}$. Note that this is numerically instable when $|x| < \epsilon$. We special case this (and its derivative) to allow for better propagation.

2. **Log Derivative.** We find that we need an explicit derivation of $D_x \log y$ for our Manifold ODE on the Sphere (see D.7). Note that this value can be computed using backpropagation, but we derive it explicitly instead due to numerical instability of the higher order derivatives of some of our functions.

Lemma. *For $x, y \in \mathcal{S}^n$, and $r = x^T y$, if $|r| \neq 1$, then*

$$D_y \log_x(y) = \left(\frac{r \arccos(r)}{(1-r^2)^{3/2}} - \frac{1}{1-r^2} \right) (y - rx) x^T + \frac{\arccos(r)}{\sin(\arccos(r))} \left(I - xx^T \right) \tag{28}$$

The limit of $D_y \log_x(y)$ as $r = x^T y \to 1$ is $I - xx^T$.

Proof. We differentiate the equation of the logarithmic map as given in Table 3. First, suppose that $|r| \neq 1$. By the product rule we have,

$$D_y \log_x(y) = D_y \left(\frac{\arccos(r)}{\sin(\arccos(r))} \right) (y - rx) + \frac{\arccos(r)}{\sin(\arccos(r))} D_y(y - rx).$$

The summand on the right is given by

$$\frac{\arccos(r)}{\sin(\arccos(r))} (I - xx^T).$$

To compute the left summand, we use the chain rule and differentiate by r

$$D_y \left(\frac{\arccos(r)}{\sin(\arccos(r))} \right) (y - rx) = \frac{\partial}{\partial r} \left(\frac{\arccos(r)}{\sin(\arccos(r))} \right) (y - rx) x^T$$

$$= \left(\frac{r \arccos(r)}{(1-r^2)^{3/2}} - \frac{1}{1-r^2} \right) (y - rx) x^T$$

To check that the limit as $x^T y \to 1$ is $I - xx^T$, it is enough to compute three separate limits that are all finite. It is clear that

$$\lim_{y \to x} (y - rx) x^T = 0.$$

Since the limit of the other term in the left summand can be shown to be finite, this means that the left summand is zero in the limit. For the right summand, the only term that depends on y has a limit

$$\lim_{r \to 1} \frac{\arccos(r)}{\sin(\arccos(r))} = \lim_{r \to 1} \frac{\arccos(r)}{\sqrt{1-r^2}} = 1$$

where the final equality can be seen by L'Hopital's rule. Thus, the entire limit is $I - xx^T$. $\qquad \square$

D.6. Backpropagation

To update the parameters of an MCNF, we need to differentiate $\log p(\text{MODE})$ with respect to θ, so we need to differentiate each of the summands in (3) with respect to θ. Differentiating through neural ODE blocks is done with the Euclidean adjoint method (Pontryagin et al., 1962; Chen et al., 2018; Grathwohl et al., 2019), which, for a loss L depending on the solution $\mathbf{y} : [t_s, t_e] \to \mathbb{R}^n$ to a differential equation with dynamics $g(\mathbf{y}(t), t; \theta)$, gives that

$$\frac{\partial L}{\partial \theta} = - \int_{t_e}^{t_s} \frac{\partial L}{\partial \mathbf{y}(t)} \frac{\partial g(\mathbf{y}(t), t; \theta)}{\partial \theta} \, dt \tag{29}$$

Differentiating the dynamics is done with standard backpropagation. The adjoint state $\frac{\partial L}{\partial \mathbf{y}(t)}$ is computed by the solution of the adjoint differential equation (2) for Euclidean space, with initial condition $\frac{\partial L}{\partial \mathbf{y}(t_e)}$. The derivative of the loss $\frac{\partial L}{\partial \mathbf{y}(t_e)}$ can be computed directly. For MCNF, L is taken to be the negative log likelihood.

For the hyperbolic and spherical cases, the log determinant terms take simple forms (as in equations 25 and 27), and are thus easy to differentiate through. Moreover, for the hyperbolic VAE models, we train by maximizing the evidence lower bound (ELBO) on the log likelihood, so that differentiation is done with a reparameterization as in Nagano et al. (2019).

D.7. Designing Neural Networks

D.7.1. Construction

In general we construct the dynamics $f(\mathbf{z}(t), t) \in T_{\mathbf{z}(t)}\mathcal{M}$ as follows. Suppose $\mathcal{M}$ is embedded in some $\mathbb{R}^d$. Then we construct f to be a neural network with input of dimension $d + 1$. The first d values are the manifold input and the last value is the time. The output of the neural network will be some vector in $\mathbb{R}^d$. To finalize, we project onto the tangent space using the linear projection $\text{proj}_{\mathbf{z}(t)}$.

Hyperbolic Space. Since hyperbolic space $\mathbb{H}^n$ is diffeomorphic to Euclidean space, we can parameterize all manifold dynamics with a corresponding Euclidean dynamic with an $\exp_{\mu_0}$, where μ_0 is the point $(1, 0, \ldots, 0) \in \mathbb{R}^{n+1}$. In general since we only require one chart, our Manifold ODE consists of $\exp_0 \circ \text{ODE} \circ \log_0$, which means that we can model our full dynamics only in the tangent space (not on the manifold). By picking our basis, we can represent elements of $T_0 \mathcal{M}$ as element of $\mathbb{R}^n$. For the ODE block, we parameterize using a neural network f which takes in an input of dimension $n + 1$ (which is a tangent space element and time) and outputs an element of dimension n.

Spherical Space.

For the spherical case, we use the default construction (with projection), as there is no global diffeomorphism. Note that when passing from the manifold to tangent space dynamics, we require $D_y \log_x$. We also must invoke a radius of injectivity, as opposed to hyperbolic space. This is π for all points.

D.7.2. Existence of a Solution

We construct our networks in such a way that the Picard-Lindelöf theorem holds. Our dynamics are given by $D_z \varphi \circ f$ where φ is a chart and f is a neural network. These are well behaved since 1) the neural network dynamics are well behaved using tanh and other Lipchitz nonlinearities and 2) the chart is well behaved since we can bound the domain to be compact.

E. Experimental Details

E.1. Data

In our code release, we will include functions to generate samples from the target densities that are used for density estimation in section 5.

Hyperbolic Density Estimation We detail the hyperbolic densities in each row of Figure 2.

1. The hyperbolic density in the first row of Figure 2 is a wrapped normal distribution with mean $(-1, 1)$ and covariance $\frac{3}{4}I$.

2. The second density is built from a mixture of 5 Euclidean gaussians, with means $(3, 0), (-3, 0), (0, 3), (0, -3)$, and

$(0, 0)$, and covariance $\frac{1}{2}I$. The resulting hyperbolic density is obtained by viewing $\mathbb{R}^2$ as the tangent space $\mathcal{T}_0\mathcal{M}$, and then projecting the Euclidean density to the Hyperboloid by $\exp_0$.

3. The third density is a projection onto the hyperboloid of a uniform checkerboard density in $\mathcal{T}_0\mathcal{M}$. The square in the second row and third column of the checkerboard has its lower-left corner at the origin $(0, 0)$. Each square has side length 1.5.

4. The fourth density is a mixture of four wrapped normal distributions. Letting $s = 1.3$, $\sigma_1 = .3$ and $\sigma_2 = 1.5$, the wrapped normals are given as:

$$\mathcal{G}\left((0, s, s), \operatorname{diag}(\sigma_1, \sigma_2)\right) \qquad \mathcal{G}\left((0, -s, -s), \operatorname{diag}(\sigma_1, \sigma_2)\right)$$
$$\mathcal{G}\left((0, -s, s), \operatorname{diag}(\sigma_2, \sigma_1)\right) \qquad \mathcal{G}\left((0, s, -s), \operatorname{diag}(\sigma_2, \sigma_1)\right)$$

Spherical Density Estimation Details are given about the densities that were learned in each row of Figure 3.

1. The density in the first row of Figure 3 is a wrapped normal distribution with mean $\frac{1}{\sqrt{3}}(-1, -1, -1)$ and covariance $\frac{3}{10}I$.

2. The second density is built from a mixture of 4 wrapped normals, with means $\frac{1}{\sqrt{3}}(1, 1, 1)$, $\frac{1}{\sqrt{3}}(-1, -1, -1)$, $\frac{1}{\sqrt{3}}(-1, -1, 1)$, and $\frac{1}{\sqrt{3}}(1, 1, -1)$; all components of the mixture have covariance $\frac{3}{10}I$.

3. The third density is a uniform checkerboard density in spherical coordinates $(\varphi, \theta) \in [0, 2\pi] \times [0, \pi]$. The rectangle in the second row and third column of the checkerboard has its lower-left corner at $(\pi, \pi/2)$. The side length of each rectangle in the φ-axis is $\pi/2 - 0.2$, and the side length in the θ-axis is $\pi/4 - 0.1$.

Variational Inference For variational inference, we dynamically binarize the MNIST and Omniglot images with the same procedure as given in Skopek et al. (2019). We resize the Omniglot images to 28×28, the same size as the MNIST images.

E.2. Density Estimation

In section 5 we train on batches of 200 samples from the target density (or batches of size 100 for the discrete spherical normalizing flows (Rezende et al., 2020)). Our MCNF models and the hyperbolic baselines use at most 1,000,000 samples, with early stopping as needed—the hyperbolic baselines sometimes diverge when training for too many batches. As suggested in Rezende et al. (2020), we find that the spherical baseline does indeed needed more samples than this (at least 5,000,000), so we allow it to train until the density converges. Although we do not investigate sample efficiency in detail, we find that our MCNF is able to achieve better results than the spherical baseline with, frequently, over an order of magnitude fewer samples than the spherical baseline. Note that all methods use the Adam optimizer (Kingma & Ba, 2014). For our MCNF, our dynamics are given by a neural network of hidden dimension 32 and 4 linear layers with tanh activation; for each integration we use a Runge-Kutta 4 solver.

Hyperbolic Normalizing Flows For the hyperbolic discrete normalizing flows, we train with 4 hidden blocks, hidden flow dimension of 32, and tanh activations. The prior distributions used are given in section D.3 and target distributions are given in section E.1.

Spherical Normalizing Flows For the discrete spherical normalizing flows from Rezende et al. (2020), we use the recursive flow for $\mathbb{S}^2$. In designing this flow, we use the non-compact projection (NCP) flow for $\mathbb{S}^1$ and the autoregressive spline flow from Durkan et al. (2019) for the interval $[-1, 1]$. To increase expressiveness for the $\mathbb{S}^1$ flow, we consider learning a convex combination of NCP flows over the circle. Let the number of components in this combination be n. To increase the expressiveness of the spline flow over $[-1, 1]$, we increase the number of segments. Let the number of segments be k. We tuned n and k for each of the 3 spherical densities to yield the best results. Note that the best n and k frequently ended up being fairly small for the more simple densities, since having an overly expressive model for simple densities ended up being hard to train and produced undesirable artifacts.

The prior distributions used for all target distributions are given in section D.3. For the first spherical target distribution in section E.1, we use $k = 2$ and $n = 1$. For the second spherical target distribution we use $k = 6$ and $n = 2$. For the final spherical target distribution we use $k = 32$ and $n = 12$.

E.3. Variational Inference

In each model for variational inference, we train with a hyperbolic or Euclidean VAE. Following Bose et al. (2020), in both cases, the mean and variance encoders are taken to be one layer neural networks with a hidden dimension of 600. The ReLU nonlinearity is used for the hyperbolic VAE and the LeakyReLU is used for the Euclidean VAE. As is often done, we take the weight matrices of the first linear layer to be shared for the mean and variance (Bose et al., 2020; Kingma & Welling, 2014). The decoder is taken to have a symmetric architecture, with the input coming from the latent space and the output being a decoded image. We vary the latent dimension from 2 to 6 in our experiments. When we add a discrete normalizing flow, we use 2 hidden blocks and 2 hidden dimensions per block with a hidden layer of size 128 and tanh activations, again following Bose et al. (2020). For continuous flows, we replace this with a neural ODE or MCNF block where the dynamics are parameterized by a two-layer network of hidden size 128 with tanh activations. For numerical integration we use the Runge-Kutta 4 solver.

F. Dynamic Chart Method

Here we elaborate on the dynamic chart method presented in Section 3 of our paper. Specifically, we discuss the significance of dynamic charts and the generality of the multi-chart MCNF (which allows learning of arbitrary densities on manifolds with conjugate points, like $\mathbb{S}^2$).

F.1. Choosing Charts

The Benefit of Dynamic Charts. Note that our dynamic chart trivialization is performed in the main paper simply by splitting the time interval $[t_s, t_e]$ up uniformly into segments of length ϵ and learning the solution locally via exp-map charts at the anchor points (endpoints of the segments). Such a splitting allows for the *ball of injectivity* (induced by the radius of injectivity) to "move" throughout the training process, so that it always surrounds the locally relevant region (centered at the anchor point). This approach allows for transportation of mass that evades the conjugate point problem (which Gemici et al. (2016) does not allow for).

This becomes especially clear if we consider the case of conjugate points on a sphere, i.e. the case of antipodal points. Consider a task in which we have a prior with mass surrounding one antipodal point and the target density is concentrated around the other point. A one chart approach would have trouble transporting the mass due to the fact that a fixed exp-map can never have a ball of injectivity (induced by the radius of injectivity π, in this case) that includes both points throughout training. However, our dynamic approach allows this ball of injectivity to shift throughout the training process and makes correct transportation of mass for such a scenario possible. An experiment testifying to this is given in Appendix F.2 (and Figure 5).

Non-uniform Time-domain Segmenting. While our approach allows for the ball of injectivity to shift, the dynamic chart method is not limited to a uniform ϵ-segment splitting of the $[t_s, t_e]$ interval. Similar benefits may be derived with an alternative splitting. However, it is not clear what additional benefits a non-uniform splitting might bring without prior knowledge of local manifold topology. Our experiments find that a uniform split works well for the densities we tested.

Control of Local Dynamics Note that although the dynamic chart approach makes it possible for the ball of injectivity to move (making it possible to learn general densities), local dynamics may still cause issues. This is because the local dynamics of the ODE may cause the solver to venture to the edges of the ball of injectivity surrounding the current anchor point x, thereby causing instability. One may resolve this issue by enforcing a Lipschitz constraint on the solution by explicitly bounding the derivative (e.g. bounding $\frac{d\mathbf{y}(t)}{dt}$). If we call the Lipschitz constant $L \in \mathbb{R}$, the length of chart domain $\epsilon > 0$, and the radius of injectivity r_x at the current anchor point x, we would want to enforce the constraint such that $\epsilon L < r_x$, i.e. $L < r_x/\epsilon$. Notice that this is only a local Lipschitz constraint, since it depends on the radius of injectivity at each anchor point, and moreover, on the way the time-domain segmenting is done. Enforcing this constraint, even in a simplistic way (e.g. thresholding $\frac{d\mathbf{y}(t)}{dt}$ at r_x/ϵ), ensures that the dynamics are configured so the local solution stays within the ball of injectivity and instability is avoided. We note that for most of our experiments, we did not worry about this, as our method was already stable, but we include this clarification in the case that it becomes necessary to maintain stability (for e.g. a particularly complicated density on a manifold with conjugate points).
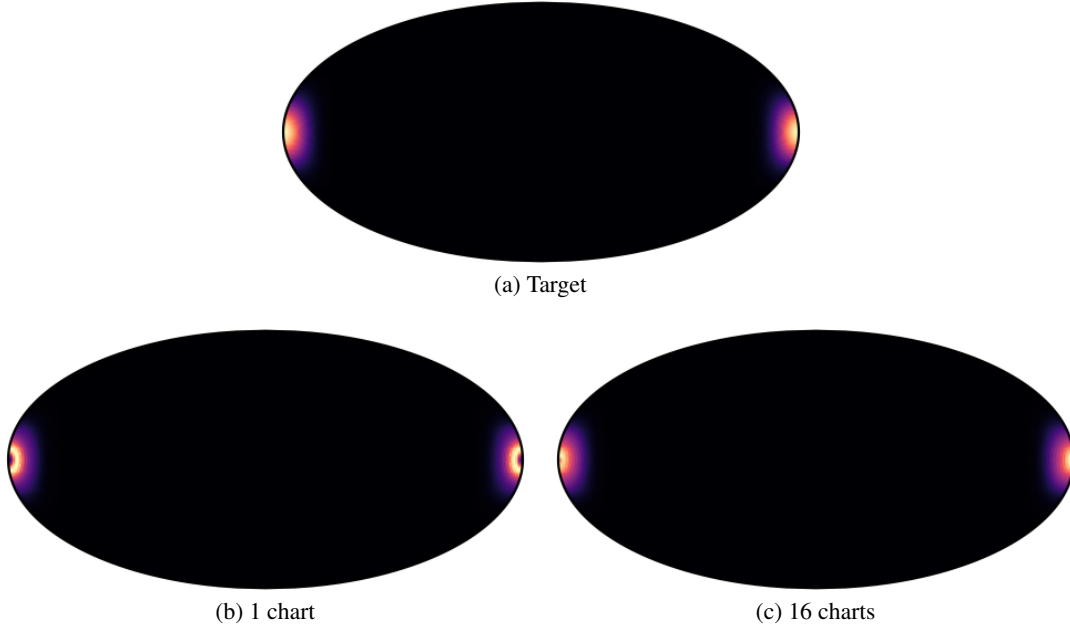
(a) Target

(b) 1 chart

(c) 16 charts

Figure 5. Comparison of MCNF for different numbers of charts. Note that using just 1 chart is not enough to learn the density concentrated at the antipode, while the 16 chart model learns the density well.

F.2. Expressivity and Generality of MCNF

To demonstrate the effectiveness of our dynamic chart method in getting around conjugate points and numerical instability around them, we set up a particular density on the sphere for estimation. The target density is a $\mathrm{vMF}((1,0,0),30)$ distribution, which is heavily concentrated around $(1,0,0)$, as shown in Figure 5 (a). We still take our base distribution as the vMF centered at $\mu_0 = (-1,0,0)$ (see Figure 4), but now we take it to be more concentrated by setting $\kappa = 3$. To learn the target density using this base distribution, our MCNF must learn to move probability density from samples around $(1,0,0)$ to areas of high base probability density around the antipodal point $(-1,0,0)$.

As shown in Figure 5, MCNF with just one chart is not able to learn a density with high probability at $(1,0,0)$, as it is unable to move samples around $(1,0,0)$ close enough to the antipodal point. On the other hand, MCNF with 16 charts is able to do so, thus validating our model's ability to get around conjugate points with the dynamic chart method.

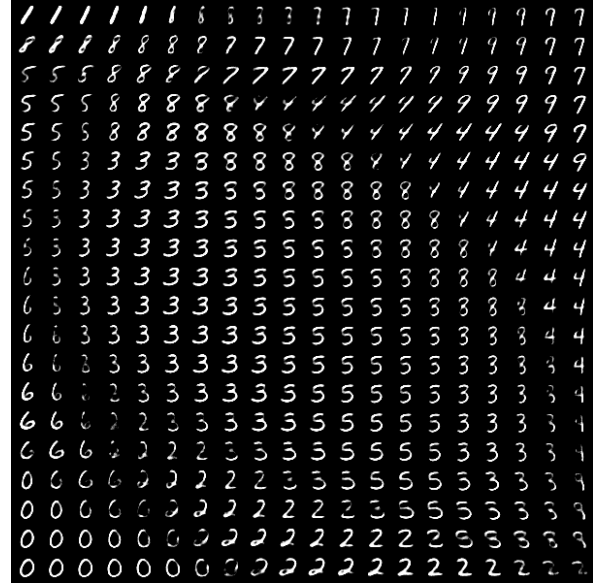## G. Generated MNIST Samples

Here, we generate samples from MNIST using our trained hyperbolic MCNF. To do this, we use an MCNF with the same setup as in E.3, except with a slightly larger VAE architecture. We add a linear layer to both the mean and variance networks, add an additional shared linear layer for both of them, and add a linear layer to the decoder. MNIST samples generated with our approach are given in Figure 6. With a latent space of dimension 2, the MCNF generates examples that resemble real digits. Interpolating between points in the latent space gives hybrid intermediaries that meaningfully represent semantic change (for instance, going between a "4" and a "7" produces instances of "9").
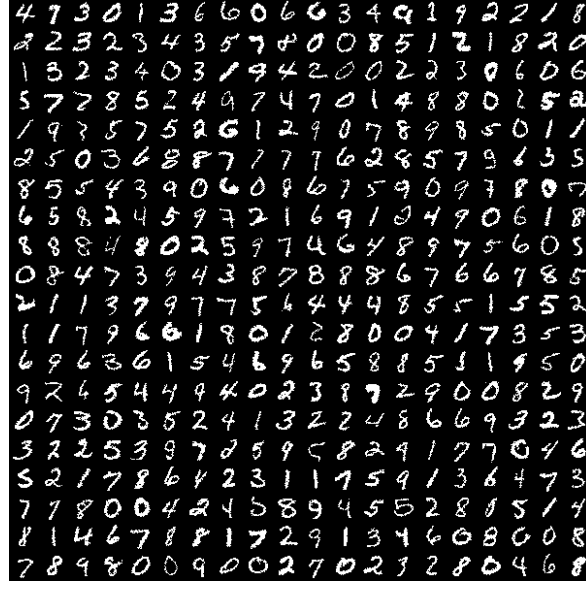
(a)
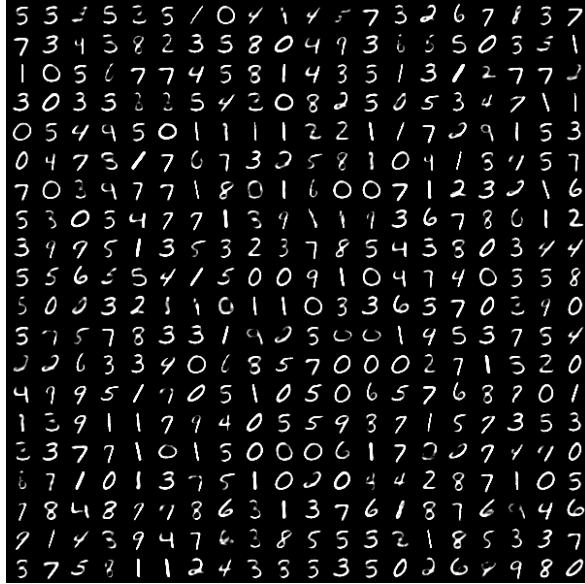
(b)

(c)

Figure 6. (a) Real sample images from (Binarized) MNIST. (b) Random generated samples from a hyperbolic MCNF trained on Binarized MNIST with latent dimension 2. (c) Generated samples from the same trained MCNF, where the latent variables are taken from the projection onto $\mathbb{H}^2$ of a uniformly spaced grid on the tangent space $\mathcal{T}_0\mathbb{H}^2$. The generated samples are visualized in their corresponding positions on $\mathcal{T}_0\mathbb{H}^2$.