Augmented Generalized Meanflows for Few Step Generation and Evaluation

Justin Lin

December 2025

School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213

COMMITTEE
Zico Kolter

Submitted in partial fulfillment of the requirements
for the Senior Honors Thesis

1 Abstract

We introduce a framework for learning augmented Riemannian flow maps that enables both fast sampling and tractable likelihood evaluation on curved manifolds. Existing Riemannian flow-based generative models rely on discretizing continuous-time geodesic or projected ODEs/SDEs and therefore require hundreds to thousands of NFEs to obtain low-bias samples and accurate log-density estimates. In contrast, our method jointly learns (1) a tangent-space-consistent velocity field, (2) its metric-dependent divergence, and (3) an augmented correction flow that compensates for curvature-induced model error. This allows us to approximate the exact Riemannian flow map in only a few integration steps.

To further reduce discretization bias, we incorporate a Riemannian rejection sampling procedure built from the instantaneous divergence and the learned log-Jacobian correction. This mechanism yields state-of-the-art likelihoods and sample quality with orders-of-magnitude fewer NFEs compared to prior manifold-aware flows.

Fast and accurate generative modeling on Riemannian manifolds is critical in scientific settings—including molecular conformer generation, protein and drug design, directional statistics, and geometric representation learning—where probability densities evolve on non-Euclidean domains. Our framework provides a scalable, geometry-consistent approach for generative modeling and likelihood evaluation in these regimes.

2 Introduction

Generative models have made remarkable progress in recent years, enabling high-fidelity synthesis of images, video, audio, and many other data modalities in Euclidean spaces. Diffusion models, normalizing flows, and more recent flow-matching-based approaches exhibit strong empirical performance and provide well-defined training objectives that scale to large datasets and high-dimensional problems. Despite these successes, a fundamental limitation persists: most state-of-the-art generative models implicitly assume that the underlying data manifold is flat, Euclidean, and globally parameterizable.

These geometric constraints introduce both theoretical and practical challenges. Directly applying Euclidean generative models to manifold-valued data often leads to artifacts, the violation of constraints (e.g., vectors leaving the sphere), distorted likelihoods, and sampling trajectories that fail to respect the underlying geometry. Moreover, many Euclidean results—such as score integrability, path consistency, or unbiased likelihood evaluation—do not transfer trivially to curved spaces.

At the same time, recent developments in few-step generative modeling—such as MeanFlow and Joint Distillation—have shown that Euclidean generative processes can be accelerated dramatically by leveraging analytically derived flows and carefully constructed distillation objectives. Extending such ideas to Riemannian manifolds presents new opportunities: the ability to compute likelihoods on curved spaces, to perform joint learning of forward and inverse flows, and to enable fast sampling while respecting geometric constraints. Moreover, access to accurate instantaneous divergence enables rejection sampling, adaptive noise selection, and noise-optimized training schedules, all of which can significantly improve generative model quality.

2.1 Riemannian Manifolds

A *Riemannian manifold* $(\mathcal{M}, g)$ is a smooth differentiable manifold equipped with a smoothly varying, positive-definite inner product

$$g_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R},$$

defined on each tangent space $T_x\mathcal{M}$. The metric g induces all geometric notions required for continuous-time generative modeling—distances, volumes, gradients, divergences, geodesics, and parallel transport—and thereby determines how probability densities evolve under vector fields on $\mathcal{M}$.

2.1.1 Local Coordinates and the Metric Tensor

Let $\varphi : U \subset \mathcal{M} \to \mathbb{R}^d$ be a coordinate chart with coordinates $x = (x^1, \ldots, x^d)$. The metric is represented in coordinates as a symmetric positive-definite matrix

$$G(x) = \big[g_{ij}(x)\big]_{i,j=1}^{d}, \qquad g_{ij}(x) = g_x\left(\frac{\partial}{\partial x^i}, \frac{\partial}{\partial x^j}\right).$$

This induces the norm

$$\|v\|_{g_x}^2 = v^\top G(x)\, v, \qquad v \in T_x\mathcal{M}.$$

2.1.2 Riemannian Volume Form and Change-of-Variables

The metric defines a canonical volume element

$$d\mathrm{vol}_g(x) = \sqrt{\det G(x)}\, dx,$$

which plays the role of the Lebesgue measure in Euclidean space. A probability density $p : \mathcal{M} \to \mathbb{R}_+$ is always interpreted with respect to this volume form:

$$\int_{\mathcal{M}} p(x)\, d\mathrm{vol}_g(x) = 1.$$

This implies that log-density expressions in likelihood evaluations must include the metric-determinant correction term $\frac{1}{2}\log\det G(x)$ when transforming between coordinate systems. Such corrections are crucial in extending continuous normalizing flows or flow-matching methods to curved spaces.

2.1.3 Geodesics, Exponential Map, and Logarithmic Map

The metric determines a unique Levi–Civita connection ∇, the torsion-free, metric-preserving affine connection on $T\mathcal{M}$. A geodesic $\gamma(t)$ satisfies

$$\nabla_{\dot\gamma(t)}\dot\gamma(t) = 0,$$

which in coordinates becomes

$$\ddot\gamma^k + \Gamma_{ij}^k(\gamma)\,\dot\gamma^i\dot\gamma^j = 0,$$

where the Christoffel symbols are

$$\Gamma_{ij}^k = \tfrac{1}{2}g^{k\ell}\big(\partial_i g_{j\ell} + \partial_j g_{i\ell} - \partial_\ell g_{ij}\big).$$

The *exponential map* at x is defined by

$$\exp_x(v) = \gamma_v(1),$$

where γ_v is the geodesic with initial position x and velocity v. When defined, its inverse is the *logarithmic map*

$$\log_x(y) = v \in T_x\mathcal{M} \quad \text{such that} \quad \exp_x(v) = y.$$

These maps are central to Riemannian flow matching, tangent-space parameterizations, and intrinsic ODE integrators on manifolds.

2.1.4 Gradients and Divergences on Riemannian Manifolds

For a smooth scalar function $f : \mathcal{M} \to \mathbb{R}$, the Riemannian gradient is defined by

$$g_x(\nabla f(x), v) = df_x(v), \qquad \forall v \in T_x\mathcal{M}.$$

In coordinates,

$$\nabla f(x) = G(x)^{-1} \nabla_{\text{Euclid}} f(x).$$

For a vector field $X : \mathcal{M} \to T\mathcal{M}$, the Riemannian divergence is

$$\operatorname{div}_g(X) = \frac{1}{\sqrt{\det G(x)}} \sum_{i=1}^{d} \frac{\partial}{\partial x^i} \big(\sqrt{\det G(x)}\, X^i \big).$$

This divergence governs the evolution of densities p_t under flows:

$$\frac{\partial p_t}{\partial t} = -\operatorname{div}_g(p_t X).$$

In particular, the log-likelihood along a trajectory $x(t)$ satisfies the Riemannian continuity equation

$$\frac{d}{dt} \log p_t(x(t)) = -\operatorname{div}_g X(x(t)).$$

2.1.5 Manifold-Constrained ODEs

A valid flow on $\mathcal{M}$ must satisfy $X(x) \in T_x\mathcal{M}$ for all x. Typical numerical strategies include:

- projection-based integrators (e.g. for spheres or SPD manifolds),
- intrinsic geodesic integrators using the exponential map, and
- first-order retractions approximating $\exp_x$.

The flow ODE

$$\dot{x}(t) = X(x(t)), \qquad x(t) \in \mathcal{M},$$

must be solved in such a way that $x(t)$ remains on the manifold for all t.

2.1.6 Pushforward of Probability Densities

For a diffeomorphism $\Phi : \mathcal{M} \to \mathcal{M}$, the pushforward density is

$$(\Phi_{\#} p)(y) = p(\Phi^{-1}(y)) \, \left| \det D\Phi^{-1}(y) \right|_g ,$$

where the determinant is computed with respect to the Riemannian volume form. In coordinates,

$$\log |\det D\Phi(x)| = \log |\det J_\Phi(x)| + \frac{1}{2} \log \frac{\det G(\Phi(x))}{\det G(x)}.$$

The second term is a curvature-dependent correction absent in Euclidean flows.

For continuous-time flows, integrating this divergence yields the Riemannian CNF likelihood:

$$\log p_1(x_1) = \log p_0(x_0) + \int_0^1 \operatorname{div}_g X(x(t))\, dt.$$

2.2 Few Step Generative Models

Continuous Normalizing Flows (CNFs) and Flow Matching (FM) provide two complementary formulations for constructing generative models through time-dependent vector fields. Both models specify a flow map $\varphi_{0\to 1}$ via an ordinary differential equation (ODE), but differ fundamentally in how the vector field is trained and how likelihoods are evaluated. We review each framework below.

2.2.1 Continuous Normalizing Flows

A Continuous Normalizing Flow [4] defines a generative model by transporting samples from a base distribution p_0 through the ODE

$$\frac{d}{dt} x_t = u_t(x_t), \qquad x_0 \sim p_0, \tag{1}$$

where $u_t : \mathbb{R}^d \to \mathbb{R}^d$ is a learned velocity field. The density of x_t evolves according to

$$\frac{d}{dt} \log p_t(x_t) = -\operatorname{div}\big(u_t(x_t)\big) = -\operatorname{tr}\left(\frac{\partial u_t}{\partial x_t}\right). \tag{2}$$

Integrating (2) yields the exact log-likelihood:

$$\log p_1(x_1) = \log p_0(x_0) - \int_0^1 \operatorname{div}(u_t(x_t))\, dt. \tag{3}$$

In practice, computing the divergence at every ODE step requires backpropagating through the Jacobian of u_t which can be computationally expensive as the model size scales, leading to the widely used *Hutchinson estimator*:

$$\operatorname{div}(u_t(x)) = \mathbb{E}_{\epsilon\sim D}\big[\epsilon^\top J_t(x)\epsilon\big],$$

where $J_t = \partial u_t/\partial x$ and D is a distribution which $E[\epsilon\epsilon^T] = I$. This stochastic estimator introduces variance and increases computational cost. Moreover, CNFs require numerically integrating both the flow and the divergence, often necessitating hundreds of NFEs for accurate likelihoods. Despite these limitations, CNFs provide an exact, principled likelihood formulation and motivate subsequent methods that aim to eliminate the need for score estimation or Hutchinson-based divergences.

2.2.2 Flow Matching

Flow Matching (FM) [8] provides a scalable alternative to CNFs by supervising the vector field u_t directly, without requiring likelihood gradients. Instead of matching $\nabla \log p_t(x)$ (as in score-based models), FM matches the *conditional velocity* of an analytically constructed interpolation between data and noise.

Given paired points (x_0, x_1) sampled from an optimal transport plan or other coupling, FM defines an interpolation curve $\gamma(t; x_0, x_1)$ and its derivative:

$$v_t^\star(x_0, x_1) = \frac{d}{dt}\gamma(t; x_0, x_1).$$

FM trains a model u_t via the regression loss

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,x_0,x_1}\Big[\|u_t(\gamma(t)) - v_t^\star(x_0, x_1)\|^2\Big]. \tag{4}$$

After training, sampling is performed by integrating the learned ODE $\frac{d}{dt}x_t = u_t(x_t)$ exactly as in CNFs, but the vector field is often significantly easier to learn and generalizes better.

Flow Matching yields vector fields that solve the same underlying flow as CNFs, but the training objective differs fundamentally.

2.2.3 Mean flows and generalised flow maps on Riemannian manifolds

A core limitation of standard flow-matching models is that they learn the *instantaneous* velocity field $v_t(x)$ of the probability flow ODE,

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = v_t(X_t), \qquad X_0 \sim p_0,\ X_1 \sim p_1,$$

and then approximate the time integral of this vector field at inference time using a numerical ODE solver. This induces a tight coupling between sample quality and the number of function evaluations (NFEs). Two recent lines of work address this limitation in complementary ways:

1. *Mean Flows* [7] introduce a new target field, the *average velocity*, and show that it can be learned directly via a local identity that relates average and instantaneous velocities. This yields a principled one-step (1-NFE) generative model in Euclidean space.

2. *Generalised Flow Maps (GFM)* [5] extend the Euclidean flow-map framework to arbitrary Riemannian manifolds, and show how to train geometric few-step models via self-distillation. Under appropriate design choices, they recover and generalise consistency models, shortcut models, and MeanFlows (via a Riemannian analogue dubbed *Generalised Mean Flows*).

We briefly review both constructions and their connection.

Mean Flows: average velocity and the MeanFlow identity. Let p_t denote the probability path between a prior p_0 and data p_1, and let $v_t(x)$ be the *marginal* instantaneous velocity field used in flow matching,

$$\partial_t \rho_t(x) + \nabla \cdot \big(\rho_t(x)\, v_t(x)\big) = 0,$$

with ρ_t the density of p_t. Mean Flows [7] introduce a two-time *average velocity* field $\bar{v}(x, r, t)$ for $t > r$, defined (informally) as the time-averaged displacement generated by the instantaneous velocity between times r and t:

$$\bar{v}(x_t, r, t) := \frac{1}{t-r} \int_r^t v_s\big(X_s\big)\, \mathrm{d}s, \qquad X_r \mapsto X_t, \tag{5}$$

where X_s is the trajectory of the probability flow ODE. In practice, one works with the marginal version of $\bar{v}$, analogous to the marginal velocity in flow matching, so that $\bar{v}$ is a deterministic field of (x, r, t) independent of the particular trajectory realisation. [**?**] show that this definition implies a local identity relating $\bar{v}$ and v.

Starting from $(t-r)\, \bar{v}(x_t, r, t) = \int_r^t v_s(X_s)\, \mathrm{d}s$ and differentiating with respect to t, one obtains

$$\bar{v}(x_t, r, t) + (t-r)\, \frac{\mathrm{d}}{\mathrm{d}t} \bar{v}(x_t, r, t) \;=\; v_t(x_t). \tag{6}$$

Rearranging yields the *MeanFlow identity*

$$\bar{v}(x_t, r, t) \;=\; v_t(x_t) \;-\; (t-r)\, \frac{\mathrm{d}}{\mathrm{d}t} \bar{v}(x_t, r, t), \tag{7}$$

which holds for the ground-truth fields. The total derivative decomposes as

$$\frac{\mathrm{d}}{\mathrm{d}t} \bar{v}(x_t, r, t) = \partial_t \bar{v}(x_t, r, t) + J_x \bar{v}(x_t, r, t)\, v_t(x_t),$$

where $J_x \bar{v}$ denotes the Jacobian of $\bar{v}$ with respect to x. This term can be implemented efficiently via a Jacobian–vector product (JVP) using standard automatic differentiation primitives.

MeanFlow models parameterise the average velocity with a neural network $\bar{v}_\theta(x, r, t)$ and enforce (7) in a self-consistent manner:

$$\mathcal{L}_{\mathrm{MF}}(\theta) = \mathbb{E}_{(x,t,r)} \Big[\big\| \bar{v}_\theta(x, r, t) - \mathrm{sg}\big(v_t(x) - (t-r)\, \tfrac{\mathrm{d}}{\mathrm{d}t} \bar{v}_\theta(x, r, t)\big) \big\|^2 \Big], \tag{8}$$

where $\mathrm{sg}(\cdot)$ denotes a stop-gradient operator and $v_t(x)$ is instantiated by the conditional velocity from flow matching. Crucially, the training target is computed from instantaneous velocities via local differential operations (JVPs); no time integration is required during training. At test time, one directly uses the learned average velocity for single- or few-step sampling:

$$x_r = x_t - (t-r)\, \bar{v}_\theta(x_t, r, t), \qquad \text{e.g., } x_0 = x_1 - \bar{v}_\theta(x_1, 0, 1), \tag{9}$$

benchmarks [7].

Generalised Flow Maps on Riemannian manifolds. Generalised Flow Maps (GFM) [5] start from the Riemannian probability flow equations on a manifold $(\mathcal{M}, g)$:

$$\partial_t \rho_t(x) + \mathrm{div}_g\big(\rho_t(x)\, u_t(x)\big) = 0, \qquad \frac{\mathrm{d}X_t}{\mathrm{d}t} = u_t(X_t), \tag{10}$$

where div_g is the Riemannian divergence and u_t is a tangent vector field, $u_t(x) \in T_x\mathcal{M}$. Instead of learning u_t and integrating (10) numerically, they propose to directly learn the *flow map* Φ that jumps along the probability flow ODE.

Given a Riemannian interpolant between a prior p_0 and target p_1 (constructed, e.g., by geodesic interpolation between coupled samples), GFM defines the generalised flow map $\Phi : \mathcal{M} \times \mathcal{T} \to \mathcal{M}$, $(x, r, t) \mapsto \Phi(x, r, t)$, as the unique function such that, for any solution X_t of (10),

$$\Phi(X_r, r, t) = X_t \qquad \text{for all } 0 \le r \le t \le 1. \tag{11}$$

This directly generalises the Euclidean flow-map notion of Boffi et al. [2] to the Riemannian setting. One can therefore perform few-step sampling by drawing $x_r \sim p_r$ and applying $\Phi(\cdot, r, t)$ for a small number of time pairs (r, t).

Practically, [5] parameterise Φ via a tangent vector field $w_\theta(x, r, t) \in T_x\mathcal{M}$ and the Riemannian exponential map, e.g.

$$\Phi_\theta(x, r, t) \;=\; \exp_x\big((t - r)\, w_\theta(x, r, t)\big),$$

which automatically enforces the boundary condition $\Phi_\theta(x, t, t) = x$. They show that Φ can be characterised equivalently by three conditions that generalise the Euclidean case: a *Lagrangian* condition (matching trajectories along the flow), an *Eulerian* PDE condition, and a *semigroup* condition expressing composition in time. These characterisations give rise to three self-distillation losses on manifolds:

- *Generalised Lagrangian Self-Distillation (G-LSD):* enforces that applying Φ_θ along a small time step matches the teacher trajectory induced by the implicit flow.
- *Generalised Eulerian Self-Distillation (G-ESD):* enforces the Eulerian PDE linking the derivative of Φ_θ to the instantaneous drift u_t, using Riemannian differentials and divergence.
- *Generalised Progressive Self-Distillation (G-PSD):* enforces the semigroup property by matching one large jump to two smaller jumps using geodesic distances.

Generalised Mean Flows and the connection to MeanFlow. A key observation in [5] is that the Eulerian self-distillation objective for Φ_θ yields, after placing the stop-gradient on the term containing spatial derivatives, a loss that is *closely related to a Riemannian generalisation of Mean Flows.* Concretely:

- On the *diagonal* $r = t$, the *Generalised Tangent Condition* shows that the derivative of Φ recovers the instantaneous drift: $\partial_t \Phi(x, t, t) = u_t(x)$ [5, Lemma 1]. This allows training the drift field using Riemannian Flow Matching [3].
- Off-diagonal (r, t), the G-ESD loss relates the time derivative of Φ_θ to the underlying vector field through a first-order differential identity on the manifold, implemented with Riemannian JVPs and differentials [5].
- When the flow-matching term on the diagonal is removed and only the Eulerian self-distillation part is kept, the resulting objective coincides with a *Generalised Mean Flow (G-MF)* loss, which mirrors the Euclidean MeanFlow identity but with all operations (divergence, Jacobians, time derivatives) interpreted in the Riemannian sense. This G-MF objective is trained exactly as MeanFlow, i.e. without an explicit teacher diffusion or flow model [5, 7].
- In Euclidean space, the MeanFlow identity (7) can be viewed as an Eulerian characterisation of the flow map's behaviour, and MeanFlow is a particular self-distillation scheme that learns the average velocity field induced by the flow.
- On Riemannian manifolds, GFM provides a unifying geometric framework: by choosing appropriate self-distillation losses (G-LSD, G-ESD, G-PSD) and where to apply stop-grad, one recovers consistency models, shortcut models, and (via G-MF) a Riemannian analogue of MeanFlow as special cases [5].

This perspective is particularly relevant for our setting: it suggests that one can design few-step Riemannian generative models either by directly learning average velocities (in the spirit of MeanFlow) or by learning flow maps whose Eulerian characterisations implicitly encode similar average-velocity identities, while remaining fully compatible with geometric constraints on $(\mathcal{M}, g)$.

2.3 Augmented Neural ODEs and Joint Distillation

Neural ODEs (NODEs) [4] model the evolution of a hidden state $h(t) \in \mathbb{R}^d$ by an ODE

$$\frac{d}{dt} h(t) = f_\theta(h(t), t), \qquad h(0) = x, \tag{12}$$

and interpret deep networks as continuous-depth limits of residual networks. When used as continuous normalizing flows, they also provide an ODE for the log-density via the divergence of the vector field [4]. However, these models face two core issues in our context: (i) representational limitations stemming from topological constraints of ODE flows, and (ii) the high computational cost of likelihood evaluation, which requires integrating an augmented ODE system with a divergence term. Augmented Neural ODEs and fast flow joint distillation (F2D2) address these limitations from complementary angles.

Augmented Neural ODEs. Dupont et al. [6] show that standard NODEs learn representations that are *topology-preserving*: the map $x \mapsto h(T)$ induced by an ODE flow is a homeomorphism onto its image, which implies that NODEs *cannot represent* functions that change the topology of the input space (e.g. separating two intertwined regions without tearing or gluing). They prove that this yields explicit classes of functions that NODEs cannot approximate without inducing extremely contorted flows that are hard for numerical ODE solvers to integrate reliably. :contentReference[oaicite:0]index=0

To resolve this, they introduce *Augmented Neural ODEs* (ANODEs), which lift the ODE to an augmented space $\mathbb{R}^{d+p}$:

$$\frac{d}{dt}\begin{pmatrix} h(t) \\ a(t) \end{pmatrix} = f_\theta\Big(\begin{pmatrix} h(t) \\ a(t) \end{pmatrix}, t\Big), \qquad (h(0), a(0)) = (x, 0), \tag{13}$$

where $a(t) \in \mathbb{R}^p$ are additional coordinates initialised to zero. The ODE is solved in the augmented space and only the $h(T)$ coordinates are used as features or outputs. By allowing trajectories to leave the original $\mathbb{R}^d$ subspace, ANODEs can realise non-homeomorphic maps in the original space while remaining smooth and well-behaved in the lifted space. Empirically, they find that ANODEs:

- are strictly more expressive than standard NODEs (they can represent functions NODEs cannot),
- learn *smoother, simpler flows* in the augmented space (visually closer to linear flows),
- reduce the number of function evaluations (NFEs) required by the ODE solver,
- improve stability, generalisation, and training speed on both toy and image benchmarks. :contentReference[oaicite:1]index=1

Thus, augmentation serves as a structural solution to both expressivity and computational issues: by increasing the state dimension, ANODEs make it easier for the solver to follow the flow, and therefore reduce the overall cost.

Joint Distillation as an augmented Neural ODE for sampling and likelihood. In continuous normalizing flows (CNFs), likelihood evaluation requires integrating a *joint* ODE system for both the state and its log-density:

$$\frac{d}{dt}\begin{pmatrix} x_t \\ \log p_t(x_t) \end{pmatrix} = \begin{pmatrix} v_\theta(x_t, t) \\ -\nabla v_\theta(x_t, t) \end{pmatrix}, \tag{14}$$

where v_θ is the velocity field and $div v_\theta = \mathrm{Tr}(\nabla_x v_\theta)$ its divergence [4]. Integrating this system backwards from $t = 1$ to 0 yields exact log-likelihoods, but requires: (i) fine time discretisation for accuracy and (ii) repeated divergence evaluations, often via expensive Hutchinson estimators. :contentReference[oaicite:2]index=2 As a result, likelihood evaluation typically needs hundreds to thousands of NFEs, often far more than sampling.

Ai et al. [1] observe that (14) can be viewed as an *augmented Neural ODE* in the extended state $y_t = (x_t, z_t)^\top \in \mathbb{R}^{d+1}$ with $z_t = \log p_t(x_t)$:

$$\frac{d}{dt} y_t = f(y_t, t) = \begin{pmatrix} v(x_t, t) \\ -\nabla v(x_t, t) \end{pmatrix}. \tag{15}$$

They then apply the flow-map viewpoint of Boffi et al. [**?**] to this joint system, and propose *fast flow joint distillation* (F2D2): a few-step flow-map model that simultaneously approximates the sampling trajectory and the cumulative divergence. Concretely, they parameterise a (linear) flow map for the *state* component as

$$\Phi_{X;\theta}(\hat{x}_t, t, s) = \hat{x}_t + (s - t)\, u_\theta(\hat{x}_t, t, s), \qquad u_\theta \approx \frac{1}{s-t} \int_t^s v(x_\tau, \tau)\, d\tau, \tag{16}$$

where u_θ predicts the *average velocity* between times t and s. :contentReference[oaicite:3]index=3 In the limit $s \to t$, $u_\theta(\hat{x}_t, t, t)$ recovers the instantaneous velocity.

For the *log-likelihood* component, they define an analogous flow map

$$\Phi_{Z;\theta}(\hat{x}_t, \hat{z}_t, t, s) = \hat{z}_t + (s - t)\, D_\theta(\hat{x}_t, t, s), \tag{17}$$

where

$$D_\theta(x_t, t, s) \;\approx\; -\frac{1}{s-t} \int_t^s \nabla v(x_\tau, \tau)\, d\tau \tag{18}$$

is a learned *average divergence* that depends only on the state trajectory x_τ. :contentReference[oaicite:4]index=4 Combining (16) and (17) yields a joint flow map on the augmented state $y_t = (x_t, z_t)^\top$:

$$\Phi_{Y;\theta}(\hat{y}_t, t, s) = \begin{pmatrix} \Phi_{X;\theta}(\hat{x}_t, t, s) \\ \Phi_{Z;\theta}(\hat{x}_t, \hat{z}_t, t, s) \end{pmatrix} = \hat{y}_t + (s-t)\, f_\theta(\hat{x}_t, t, s), \quad f_\theta = \begin{pmatrix} u_\theta \\ D_\theta \end{pmatrix}. \tag{19}$$

Architecturally, u_θ and D_θ share a backbone but use separate heads, reflecting the shared dependence on the underlying velocity field. :contentReference[oaicite:5]index=5

Joint distillation objective. F2D2 then defines a joint self-distillation loss that enforces: (i) the *tangent condition* (instantaneous velocity and divergence at $s = t$), and (ii) one of the flow-map characterisations (Lagrangian, Eulerian, or semigroup) for both the state and log-likelihood components. In its most general form, the objective decomposes as

$$\mathcal{L}_{\mathrm{F2D2}}(\theta) \;=\; \mathcal{L}_{\mathrm{VM}}(\theta) + \mathcal{L}_u(\theta) + \mathcal{L}_{\mathrm{div}}(\theta) + \mathcal{L}_D(\theta), \tag{20}$$

where:

- $\mathcal{L}_{\mathrm{VM}}$ enforces instantaneous velocity matching (typically via the flow-matching loss);

- $\mathcal{L}_u$ enforces a flow-map condition for the X component (e.g. semigroup or Eulerian PDE);
- $\mathcal{L}_{\text{div}}$ matches the instantaneous divergence (Z-component at $s = t$);
- $\mathcal{L}_D$ enforces a flow-map condition for the Z component, ensuring the joint map $\Phi_{Y;\theta}$ is consistent with the coupled ODE (14). :contentReference[oaicite:6]index=6

Instantiations such as Shortcut-F2D2 and MeanFlow-F2D2 specialise $\mathcal{L}_u$ and $\mathcal{L}_D$ to semigroup- and Eulerian-based self-consistency losses respectively, reusing the same framework.

3 Methods

3.1 Problem setting

Let $(\mathcal{M}, g)$ be a d-dimensional Riemannian manifold with metric tensor g_x at $x \in \mathcal{M}$. We denote by $T_x\mathcal{M}$ the tangent space at x, and by div_g the Riemannian divergence associated with the Levi–Civita connection. We assume a *base* distribution p_0 on $\mathcal{M}$ (e.g. an isotropic reference measure) and a *data* distribution p_1, and we seek a time-dependent flow that transports p_0 to p_1.

We consider a time interval $[0, 1]$ and a probability path $(p_t)_{t\in[0,1]}$ on $\mathcal{M}$ governed by a vector field

$$u : [0, 1] \times \mathcal{M} \to T\mathcal{M}, \qquad u(t, x) \in T_x\mathcal{M},$$

through the Riemannian continuity equation

$$\partial_t \rho_t(x) + div_g\big(\rho_t(x)\, u(t, x)\big) = 0, \tag{21}$$

where ρ_t is the density of p_t w.r.t. the Riemannian volume. The associated Lagrangian trajectories $(X_t)_{t\in[0,1]}$ satisfy

$$\frac{d}{dt} X_t = u(t, X_t). \tag{22}$$

3.2 Geodesic coupling

Training is performed using pairs $(x_0, x_1) \in \mathcal{M} \times \mathcal{M}$ drawn from a coupling between p_0 and p_1. For each pair we construct a Riemannian geodesic

$$\gamma_{x_0,x_1} : [0, 1] \to \mathcal{M}, \qquad \gamma_{x_0,x_1}(0) = x_0, \quad \gamma_{x_0,x_1}(1) = x_1, \tag{23}$$

and denote its velocity by

$$u_t^\star = \dot\gamma_{x_0,x_1}(t) \in T_{\gamma_{x_0,x_1}(t)}\mathcal{M}. \tag{24}$$

We view $x_t^\star := \gamma_{x_0,x_1}(t)$ and $u_t^\star$ as *geometric supervision* for the flow at time t.

For each pair we draw two times $(t, r) \in [0, 1]^2$ from a simple distribution, e.g. uniform or log-normal. We then set

$$t = \max\{s_1, s_2\}, \qquad r = \min\{s_1, s_2\},$$

and with some probability $\alpha \in (0, 1)$ we force $r = t$. The diagonal samples $t = r$ are used to impose instantaneous constraints, while off-diagonal pairs $r < t$ are used to impose finite-interval constraints.

3.3 Two-time Riemannian flow and MeanFlow loss

We parameterise a *two-time* tangent vector field

$$u_\theta : [0, 1]^2 \times \mathcal{M} \to T\mathcal{M}, \qquad (t, r, x) \mapsto u_\theta(t, r, x) \in T_x\mathcal{M}, \tag{25}$$

where t plays the role of the current time and r is an auxiliary time parameter.

On the *diagonal* $t = r$, we want $u_\theta(t, t, \cdot)$ to approximate the instantaneous velocity field $u(t, \cdot)$ of the probability flow (22). Off the diagonal $r < t$, we interpret $u_\theta(t, r, \cdot)$ as an approximation of an *average* velocity over the interval $[r, t]$ in the spirit of MeanFlow.

For a given pair (x_0, x_1) and sampled times (t, r) we write

$$x_t^\star := \gamma_{x_0,x_1}(t), \qquad u_t^\star := \dot{\gamma}_{x_0,x_1}(t).$$

We then consider the model velocity

$$u_\theta(t, r, x_t^\star) \in T_{x_t^\star}\mathcal{M}$$

and its *total derivative* with respect to t along the geodesic direction $u_t^\star$, defined as

$$\frac{d}{dt} u_\theta(t, r, x_t^\star) = \partial_t u_\theta(t, r, x_t^\star) + \nabla_{u_t^\star} u_\theta(t, r, x_t^\star), \tag{26}$$

where ∇ is the Levi–Civita connection of g. In practice, this is implemented by a Jacobian–vector product.

A MeanFlow-type identity suggests that the geodesic velocity $u_t^\star$ can be recovered from an average velocity and its time derivative. We therefore define the target

$$u_{tgt}(t, r, x_t^\star) := u_t^\star - (t - r) \frac{d}{dt} u_\theta(t, r, x_t^\star), \tag{27}$$

and minimise a Riemannian squared error between u_θ and u_{tgt}:

$$\mathcal{L}_{\mathcal{M}vel}(\theta) = \mathbb{E}_{(x_0,x_1,t,r)} \left[\frac{1}{d} \left\langle u_\theta(t, r, x_t^\star) - u_{\mathcal{M}tgt}(t, r, x_t^\star), u_\theta(t, r, x_t^\star) - u_{\mathcal{M}tgt}(t, r, x_t^\star) \right\rangle_{g(x_t^\star)} \right], \tag{28}$$

where $\mathcal{L}angle\cdot, \cdot\rangle_{g(x)}$ is the Riemannian inner product at x.

On diagonal samples $t = r$, we have $t - r = 0$ and (27) reduces to $u_{\mathcal{M}tgt}(t, t, x_t^\star) = u_t^\star$, recovering standard Riemannian flow matching.

3.4 Instantaneous divergence on a Riemannian manifold

Given a vector field $v(t,\cdot)$ on $\mathcal{M}$, its Riemannian divergence at x is

$$div_g v(t,x) = tr\big(\nabla v(t,x)\big), \tag{29}$$

where ∇v is the covariant derivative. In local coordinates, this can be written as

$$div_g v = \frac{1}{\sqrt{\det G(x)}} \sum_{i=1}^{d} \partial_{x^i}\big(\sqrt{\det G(x)}\, v^i\big), \tag{30}$$

with $G(x)$ the metric matrix and v^i the coordinate components of v.

In practice we approximate $div_g u_\theta(t,t,\cdot)$ by either:

- an *exact* trace of the Jacobian of the ambient vector field, when feasible, or
- a *Hutchinson*-type estimator $div\, v(x) \approx \mathbb{E}_\varepsilon\big[\varepsilon^\top J(x)\varepsilon\big]$, where $J(x)$ is the Jacobian of v and ε is a random Rademacher vector.

If the manifold structure provides the metric volume term $\log\det G(x)$, we correct the Euclidean divergence by

$$div_g v(x) = div\, v(x) + \frac{1}{2}\mathcal{L}angle\nabla \log\det G(x), v(x)\rangle, \tag{31}$$

where the second term is computed via a directional derivative of $\log\det G$ along v.

3.5 Divergence MeanFlow objective

In addition to the velocity field, we model a scalar function

$$D_\theta : [0,1]^2 \times \mathcal{M} \to \mathbb{R}, \qquad (t,r,x) \mapsto D_\theta(t,r,x), \tag{32}$$

which is intended to approximate a *finite-interval* quantity derived from the instantaneous Riemannian divergence $div_g u(t,x)$ over the interval $[r,t]$. For example, one can view $D_\theta(t,r,x)$ as an approximation of an average divergence

$$D_\theta(t,r,x_t^\star) \approx -\frac{1}{t-r}\int_r^t div_g u(\tau, X_\tau)\, d\tau$$

along a flow trajectory (X_τ) passing through $x_t^\star$ at time t.

We again exploit an Eulerian identity linking D_θ and the instantaneous divergence. At a given $(x_t^\star, t, r)$, consider the total derivative of D_θ w.r.t. t along a chosen tangent direction (typically the model drift $u_\theta(t,t,x_t^\star)$):

$$\frac{d}{dt} D_\theta(t,r,x_t^\star) = \partial_t D_\theta(t,r,x_t^\star) + \nabla_{u_\theta(t,t,x_t^\star)} D_\theta(t,r,x_t^\star). \tag{33}$$

We denote by

$$div^\star(t, x_t^\star)$$

a reference instantaneous divergence at $(t, x_t^\star)$, computed either from the current model or from a fixed teacher.

A MeanFlow-type relation suggests that a finite-interval divergence can be related to $div^\star$ and $\frac{d}{dt}D_\theta$ via a first-order ODE. Abstractly, we can write a target of the form

$$D_{tgt}(t, r, x_t^\star) = \Phi\Big(t, r, x_t^\star,\ \frac{d}{dt}D_\theta(t, r, x_t^\star),\ div^\star(t, x_t^\star)\Big), \tag{34}$$

for some explicit function Φ determined by the chosen identity. The concrete implementation used in our experiments is a linear combination of $(t-r)\ \frac{d}{dt}D_\theta$ and $-div^\star$.

We then minimise a mean-squared error between D_θ and D_{tgt}:

$$\mathcal{L}_{div}(\theta) = \mathbb{E}_{(x_0, x_1, t, r)}\Big[\big(D_\theta(t, r, x_t^\star) - D_{tgt}(t, r, x_t^\star)\big)^2\Big]. \tag{35}$$

In practice, we normalise D_θ and D_{tgt} with a running mean and variance to stabilise training, but this does not affect the underlying mathematical objective.

3.6 Total training objective

The overall loss combines the Riemannian MeanFlow velocity loss (28) and the divergence loss (35):

$$\mathcal{L}(\theta) = \mathcal{L}_{vel}(\theta) + \mathcal{L}_{div}(\theta). \tag{36}$$

Optionally, we employ a staged training scheme in which we first minimise only $\mathcal{L}_{\sqsubseteq\rceil\updownarrow}$ (learning the flow) and then jointly optimise $\mathcal{L}_{\sqsubseteq\rceil\updownarrow} + \mathcal{L}_{div}$ (learning both flow and divergence).

3.7 Exact and fast likelihood evaluation

Exact likelihood via augmented flow. To compute the exact log-likelihood of $x_1 \in \mathcal{M}$ under the learned flow, we consider the augmented state $(X_t, S_t) \in \mathcal{M} \times \mathbb{R}$ satisfying

$$\frac{d}{dt}\begin{pmatrix} X_t \\ S_t \end{pmatrix} = \begin{pmatrix} u(t, X_t) \\ -div_g u(t, X_t) \end{pmatrix}, \qquad X_1 = x_1,\ S_1 = 0. \tag{37}$$

Integrating (37) backward from $t = 1$ to $t = 0$ yields (X_0, S_0), and the log-density of x_1 is

$$\log p_1(x_1) = \log p_0(X_0) + S_0. \tag{38}$$

In practice this ODE is solved numerically on $\mathcal{M} \times \mathbb{R}$ using a sufficiently accurate time discretisation.

Fast few-step likelihood via two-time flow. Using the learned two-time quantities u_θ and D_θ, we can also define an approximate few-step likelihood that avoids fine temporal discretisation. For a small number of time points

$$1 = t_0 > t_1 > \cdots > t_K = 0$$

we approximate the backward flow by

$$X_{t_{k+1}} \approx X_{t_k} + (t_{k+1} - t_k)\, u_\theta(t_k, t_{k+1}, X_{t_k}), \tag{39}$$

$$S_{t_{k+1}} \approx S_{t_k} + (t_{k+1} - t_k)\, D_\theta(t_k, t_{k+1}, X_{t_k}), \tag{40}$$

starting from $(X_{t_0}, S_{t_0}) = (x_1, 0)$. After K steps we obtain X_0 and S_0, and define the approximate log-likelihood

$$\log \tilde{p}_1(x_1) = \log p_0(X_0) + S_0. \tag{41}$$

The special case $K = 1$ corresponds to a single-step approximation using $u_\theta(1, 0, \cdot)$ and $D_\theta(1, 0, \cdot)$, yielding a one-step generative model on $\mathcal{M}$.

3.8 Pop-Art Normalization for Divergence Targets

Learning divergence on Riemannian manifolds is substantially more challenging than in Euclidean settings due to the large dynamic range of the divergence signal. The instantaneous divergence of a vector field $u_\theta(t, x)$ contains contributions from both the Jacobian of the field and the manifold geometry:

$$\operatorname{div} u_\theta(t, x) = \operatorname{tr}\big(J_x u_\theta(t, x)\big) + \tfrac{1}{2} \left\langle u_\theta(t, x), \nabla \log \det g(x) \right\rangle,$$

where $g(x)$ is the Riemannian metric tensor. On curved spaces, this term can vary by several orders of magnitude across the data manifold. Naively regressing onto raw divergence values leads to instability, exploding gradients, and divergence collapse during training.

To address this issue, we employ *Pop-Art normalization* (Preserving Outputs Precisely, While Adapting Rescaling Targets), originally proposed for stabilizing value regression in reinforcement learning. Let D_{tgt} denote the unnormalized divergence supervision signal. We maintain exponential moving averages of its mean and second moment:

$$\mu_t = \beta \mu_{t-1} + (1 - \beta)\, \mathbb{E}[D_{\text{tgt}}], \qquad \sigma_t^2 = \beta \sigma_{t-1}^2 + (1 - \beta)\, \mathbb{E}[D_{\text{tgt}}^2] - \mu_t^2,$$

and construct the normalized target

$$\tilde{D}_{\text{tgt}} = \frac{D_{\text{tgt}} - \mu_t}{\sigma_t + \varepsilon}.$$

The divergence head predicts $\tilde{D}_\theta(t, x)$ and is trained via normalized regression. During likelihood computation and few-step integration, the prediction is unnormalized:

$$D_\theta(t, x) = \sigma_t\, \tilde{D}_\theta(t, x) + \mu_t.$$

Pop-Art normalization stabilizes training by:

- controlling the magnitude of divergence gradients across regions of high curvature,
- preventing divergence estimates from exploding or collapsing as training progresses,
- enabling staged training in which the velocity representation stabilizes before divergence learning,
- preserving correctness of the divergence prediction despite online rescaling.

This normalization was essential for stable joint distillation and for accurate few-step likelihood estimation across manifolds of varying dimensionality and curvature.

3.9 Rejection Sampling for Geometry-Aware Few-Step Likelihood Correction

Few-step approximations of Riemannian flow maps inevitably introduce discretization bias. Unlike in Euclidean settings, curvature induces additional distortions that accumulate during backward integration for likelihood evaluation. Moreover, the joint-distilled divergence is itself an approximation, leading to residual stochastic error in the fast log-likelihood estimator.

To mitigate these effects, we introduce a *Riemannian rejection-sampling procedure* that improves sample quality and likelihood accuracy without increasing the number of function evaluations. Given a batch of endpoints x_1, we draw K candidate base samples:

$$x_0^{(1)}, \dots, x_0^{(K)}, \qquad x_1^{(k)} = \Phi_\theta(x_0^{(k)}),$$

where Φ_θ is the learned few-step generative map. For each candidate we compute the fast approximate log-likelihood

$$L^{(k)} = \log p_0\Big(x_0^{(k)}\Big) + \int_0^1 \operatorname{div} u_\theta\Big(t, X_t^{(k)}\Big)\, dt,$$

where the divergence integral is approximated using the F2D2 estimator. These values serve as a ranking signal.

We then select either:

- the candidate with maximal approximate likelihood

$$k^\star = \arg\max_k L^{(k)},$$

- or a stochastic choice proportional to $\exp(L^{(k)})$.

This discrete correction step acts analogously to a Metropolis–Hastings update, but adapted to deterministic Riemannian flows and fast divergence estimation. It significantly reduces approximation error introduced by few-step integration and improves performance without additional NFEs.

The rejection sampler provides several benefits:

- **Bias reduction**: mitigates curvature-induced errors in few-step flow integration.
- **Enhanced sample fidelity**: improbable flow artifacts are discarded automatically.

Empirically, we observe that this geometric rejection-sampling scheme produces substantial improvements across all evaluated datasets, especially in regions where curvature or density sharpness pose challenges for few-step flow models.

4 Results

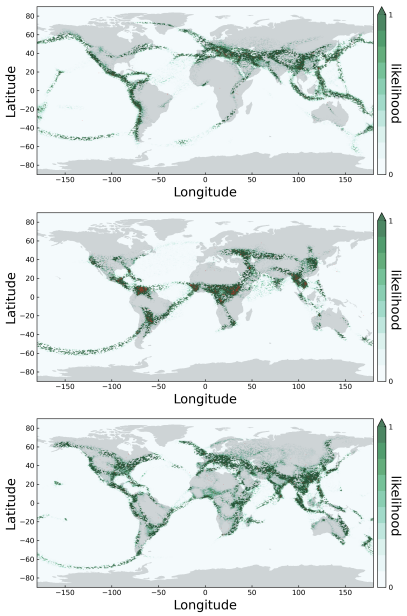

Figure 1: Caption

Dataset size	**Volcano** (827)	**Earthquake** (6,120)	**Flood** (4,875)	**Fire** (12,809)
Mixture of Kent†	-0.80 ± 0.47	0.33 ± 0.05	0.73 ± 0.07	-1.18 ± 0.06
Riemannian CNF† (Mathieu & Nickel, 2020a)	-0.97 ± 0.15	0.19 ± 0.04	0.90 ± 0.03	-0.66 ± 0.05
Moser Flow† (Rozen et al., 2021)	-2.02 ± 0.42	-0.09 ± 0.02	0.62 ± 0.04	-1.03 ± 0.03
Stereographic Score-Based†	-4.18 ± 0.30	-0.04 ± 0.11	1.31 ± 0.16	0.28 ± 0.20
RSGM† (De Bortoli et al., 2022)	-5.56 ± 0.26	-0.21 ± 0.03	0.52 ± 0.02	-1.24 ± 0.07
RDM† (Huang et al., 2022b)	-6.61 ± 0.97	-0.40 ± 0.05	0.43 ± 0.07	-1.38 ± 0.05
RFM (Chen & Lipman, 2024)	$\mathbf{-7.93 \pm 1.67}$	-0.28 ± 0.08	0.42 ± 0.05	-1.86 ± 0.11
G-LSD	-4.96 ± 0.68	-0.93 ± 0.01	-0.38 ± 0.33	-2.14 ± 0.42
G-PSD	-3.50 ± 0.22	-0.63 ± 0.13	-0.76 ± 0.13	$\mathbf{-2.48 \pm 0.71}$
G-ESD	-4.49 ± 0.20	-0.67 ± 0.08	$\mathbf{-0.88 \pm 0.38}$	-2.29 ± 0.08
G-MF	-3.73 ± 0.41	$\mathbf{-1.08 \pm 0.09}$	-0.72 ± 0.11	-2.24 ± 0.30
G-JointMF	-2.15	-0.25	0.2	-1.13
G-JointMF-Rejection Sampling				

Table 1: Comparison of log-likelihood performance across four natural disaster datasets. † indicates baseline models reported from prior work. Bold numbers denote best performance per column among all methods.

5 Comparison With Prior Work

6 Future Work

6.1 Architecture

6.1.1 Incorporating High-Capacity Architectures on Riemannian Manifolds

Current experiments primarily employ MLP-based vector fields for clarity and interpretability. However, scaling generative modeling on manifolds to high-dimensional settings—such as CLIP embeddings on S^d, molecular conformation spaces, diffusion MRI orientation distributions, or spherical image domains—will ultimately require significantly more expressive architectures. Two promising directions are Riemannian extensions of U-Nets and Diffusion Transformers (DiTs).

6.1.2 Riemannian U-Nets

U-Nets have shown exceptional performance in Euclidean diffusion models due to their hierarchical multiscale structure, skip connections, and ability to aggregate both local and global features. Extending U-Nets to manifold-valued data requires several geometric modifications:

- **Manifold-aware convolutional operators**, including:
  - tangent–space convolutions via logarithmic maps,
  - spherical convolutions (e.g., harmonic transforms or HEALPix-style discretizations),
  - mesh convolutions for discretized manifolds.

- **Projection layers** that map intermediate outputs back to the tangent space:
$$\tilde{u}(x) \;\mapsto\; \Pi_{T_x\mathcal{M}}(\tilde{u}(x))\,.$$
- **Scale-consistent positional embeddings**, ensuring temporal and geometric encodings remain coordinate-independent and respect the manifold metric.

Embedding a U-Net inside the MeanFlow or F2D2 framework would enable learning high-capacity average-velocity and divergence fields, potentially supporting high-resolution generative modeling directly on curved spaces.

6.1.3 DiT-Style Transformer Architectures

Diffusion Transformers (DiTs) have recently demonstrated state-of-the-art results in Euclidean diffusion modeling. A Riemannian analogue would require geometry-aware architectural components, such as:

- **Tokenization of manifold points** using local tangent-space patches or learned chart embeddings.
- **Attention mechanisms respecting the manifold metric**, for example:
$$\mathrm{Attn}(x_i, x_j) \;\propto\; \exp\!\left(-\frac{d_{\mathcal{M}}(x_i, x_j)^2}{\sigma^2}\right).$$
- **Geodesic-relative positional encodings** rather than Euclidean offsets, ensuring that positional structure is invariant under chart choices.

DiT architectures offer near-linear scalability in depth and extremely high representational capacity, which could dramatically improve modeling performance in complex manifold domains (e.g., torsion-angle flows on product manifolds for molecular generation).

6.2 Lipschitz Control via Spectral Normalization

For Riemannian flow models, the learned vector field $u_\theta(t, x)$ must satisfy a Lipschitz condition

$$\|u_\theta(t, x) - u_\theta(t, y)\|_{g(x)} \;\le\; L\, d_{\mathcal{M}}(x, y),$$

to ensure existence and uniqueness of flow trajectories and to guarantee numerical stability for both integration and likelihood computation. High-capacity architectures such as U-Nets and DiTs, however, can have extremely large effective Lipschitz constants due to deep compositions, skip connections, and attention mechanisms. A promising direction for future work is to enforce Lipschitz regularity through *spectral normalization*.

Spectral Normalization for Convolutional Layers. For a linear operator W acting on Euclidean space, its Lipschitz constant is given by its spectral norm $\|W\|_2$. Spectral normalization rescales each layer as

$$\hat{W} = \frac{W}{\|W\|_2},$$

which ensures that each convolutional block is 1-Lipschitz. In the U-Net setting, this provides explicit control over the Lipschitz constant of all downsampling, upsampling, and skip-connected pathways. Although the overall Lipschitz constant is the sum of the constants across all parallel paths, bounding each constituent operator prevents exponential growth in the global bound.

Spectral Normalization for Attention. Attention layers pose a greater challenge: for queries Q, keys K, and values V, the attention mechanism is

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V.$$

The softmax operator has a Lipschitz constant that grows exponentially in $\|QK^\top\|$, which can destabilize ODE or MeanFlow-based training when used as a velocity field. Applying spectral normalization to Q, K, and V,

$$Q \leftarrow \frac{Q}{\|Q\|_2}, \qquad K \leftarrow \frac{K}{\|K\|_2}, \qquad V \leftarrow \frac{V}{\|V\|_2},$$

bounds the magnitude of attention logits, thereby implicitly controlling the Lipschitz constant of the entire attention block. This is especially important when adapting DiT architectures to Riemannian manifolds, where unstable derivatives directly affect divergence estimation.

Interaction with Riemannian Geometry. On a manifold, controlling Lipschitz constants requires compatibility with the metric tensor:

$$\|W\|_g = \sup_{v \neq 0} \frac{\|Wv\|_g}{\|v\|_g}.$$

When using tangent-space convolutions or attention in coordinate charts, spectral normalization ensures that the network is Lipschitz with respect to the chart metric. Combined with projection operators $\Pi_{T_x\mathcal{M}}$, this enforces stability of the global flow under curvature, which is critical for:

- preventing exploding JVPs in MeanFlow and joint distillation,
- stabilizing backward integration for likelihood evaluation,
- ensuring geodesic consistency when projecting intermediate states back to $T_x\mathcal{M}$.

Outlook. Spectral normalization provides a principled and computationally tractable way to bound the Lipschitz constant of high-capacity networks on Riemannian manifolds. Future work includes extending existing Lipschitz-attention methods, developing manifold-aware spectral norms for convolution on curved domains, and investigating how Lipschitz constraints interact with F2D2 divergence learning and few-step likelihood estimators.

References

[1] Xinyue Ai, Yutong He, Albert Gu, Ruslan Salakhutdinov, J Zico Kolter, Nicholas Matthew Boffi, and Max Simchowitz. Joint distillation for fast likelihood evaluation and sampling in flow-based models, 2025.

[2] Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation, 2025.

[3] Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries, 2024.

[4] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2019.

[5] Oscar Davis, Michael S. Albergo, Nicholas M. Boffi, Michael M. Bronstein, and Avishek Joey Bose. Generalised flow maps for few-step generative modelling on riemannian manifolds, 2025.

[6] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes, 2019.

[7] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling, 2025.

[8] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023.