Scaling Riemannian Diffusion Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Riemannian diffusion models draw inspiration from standard Euclidean space diffusion models to learn distributions on general manifolds. Unfortunately, the additional geometric complexity renders the diffusion transition term inexpressible in closed form, so prior methods resort to imprecise approximations of the score matching training objective that degrade performance and preclude applications in high dimensions. In this work, we reexamine these approximations and propose several practical improvements. Our key observation is that most relevant manifolds are symmetric spaces, which are much more amenable to computation. By leveraging and combining various ansätze, we can quickly compute relevant quantities to high precision. On low dimensional datasets, our correction produces a noticeable improvement and is competitive with other techniques. Additionally, we show that our method enables us to scale to high dimensional tasks on nontrivial manifolds, including $SU(n)$ lattices in the context of lattice quantum chromodynamics (QCD). Finally, we apply our models to contrastively learned hyperspherical embeddings, curbing the representation collapse problem in the projection head and closing the gap between theory and practice.

1 Introduction

By learning to faithfully capture high-dimensional probability distributions, modern deep generative models have transformed countless fields such as computer vision [19] and natural language processing [42]. However, these models are built primarily for geometrically simple data spaces, such as Euclidean space for images and discrete space for text. For many applications such as protein structure prediction [13], contrastive learning [12], and high energy physics [6], the support of the data distribution is instead a Riemannian manifold such as the sphere or torus. Here, naïvely applying a standard generative model on the ambient results in poor performance as it doesn't incorporate the geometric inductive bias and can suffer from singularities [7].

As such, a longstanding goal within Geometric Deep Learning has been the development of principled, general, and scalable generative models on manifolds [8, 18]. One promising method is the Riemannian Diffusion Model [4, 25], the natural generalization of standard Euclidean space score-based diffusion models [47, 49]. These learn to reverse a diffusion process on a manifold–in particular, the heat equation–through Riemannian score matching methods. While this approach is principled and general, it is not scalable. In particular, the additional geometric complexity renders the denoising score matching loss intractable. Because of this, previous work resorts to inaccurate approximations or sliced score matching [48], but these degrade performance and can't be scaled to high dimensions. We emphasize that this fundamental problem causes Riemannian Diffusion Models to fail for even trivial distributions on high dimensional manifolds, which limits their applicability to relatively simple low-dimensional examples.

In our work, we propose several improvements to Riemannian Diffusion Models to stabilize their performance and enable scaling to high dimensions. In particular, we reexamine the heat kernel [21], which is the core building block for the denoising score matching objective. To enable denoising score matching, one needs to be able to sample from and compute the gradient of the logarithm of the heat kernel efficiently. This can be done trivially in Euclidean space as the heat kernel is a Gaussian distribution, but doing this is effectively intractable for general manifolds. By restricting our analysis to Riemannian symmetric spaces [9, 29], which are a class of a manifold with a special symmetry structure, we can make substantial improvements. We leverage this additional structure to quickly and precisely compute heat kernel quantities, allowing us to scale up Riemannian Diffusion Models to high dimensional real-world tasks. Furthermore, since almost all manifolds that practitioners work with are (or are diffeomorphic to) Riemannian symmetric spaces, our improvements are generalizable and not task-specific. Concretely our contributions are:

- **We present a generalized strategy for numerically computing the heat kernel on Riemannian symmetric spaces in the context of denoising score matching.** In particular, we adapt known heat kernel techniques to our more specific problem, allowing us to quickly, accurately, and stably train with denoising score matching.
- **We show how to exactly sample from the heat kernel using our quick computations above.** In particular, we show how our exact heat kernel computation enables fast simulation free techniques on simple manifolds. Furthermore, we develop a sampling method based on the probability flow ODE and show that this can be quickly computed with standard ODE solvers on the maximal torus of the manifold.
- **We empirically demonstrate that our improved Riemannian Diffusion Models improve performance and scale to high dimensional real world tasks.** For example, we can faithfully learn Wilson action on 4×4 $SU(3)$ lattices (128 dimensions). Furthermore, when applied to contrastively learned hyperspherical embeddings (127 dimensions), our method enables better model interpretability by recovering the collapsed projection head representations. To the best of our knowledge, this is the first example where differential equation-based manifold generative models have scaled to real world tasks with hundreds of dimensions.

2 Background

2.1 Diffusion Models

Diffusion models on $\mathbb{R}^d$ are defined through stochastic differential equations [24, 46, 49]. Given an initial data distribution p_0 on $\mathbb{R}^d$, samples $\mathbf{x}_0 \sim \mathbb{R}^d$ are perturbed with a stochastic differential equation[33]

$$\mathrm{d}\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{B}_t \tag{1}$$

where $\mathbf{f}$ and g are fixed drift and diffusion coefficients, respectively. The time varying distributions p_t (defined by $\mathbf{x}_t$) evolves according to the Fokker-Planck Equation

$$\frac{\partial}{\partial t}p_t(\mathbf{x}) = -\operatorname{div}(p_t(\mathbf{x})\mathbf{f}(\mathbf{x}, t)) + \frac{g(t)^2}{2}\Delta_x p_t(\mathbf{x}) \tag{2}$$

and approaches a limiting distribution p_T, which is normally a simple distribution like a Gaussian $\mathcal{N}(0, \sigma_T^2 I)$ through carefully chosen $\mathbf{f}$ and g. Our SDE has a corresponding reversed SDE

$$\mathrm{d}\mathbf{x}_t = (\mathbf{f}(\mathbf{x}_t, t) - g(t)^2\nabla_x \log p_t(\mathbf{x}_t))\mathrm{d}t + g(t)\mathrm{d}\overline{\mathbf{B}}_t \tag{3}$$

which maps p_T back to p_0. Diffusion models approximate the score function $\nabla_x \log p_t(\mathbf{x})$ using a neural network $\mathbf{s}_\theta(\mathbf{x}, t)$. To do this, one minimizes the score matching loss [27], which is weighted by constants λ_t:

$$\mathbb{E}_t\mathbb{E}_{\mathbf{x}_t\sim p_t}\lambda_t \left\|\mathbf{s}_\theta(\mathbf{x}_t, t) - \nabla_x \log p_t(\mathbf{x}_t)\right\|^2 \tag{4}$$

Since this loss is intractable due to the unknown $\nabla_x \log p_t(\mathbf{x}_t)$, we instead use an alternative form of the loss. One such loss is the implicit score matching loss[27]:

$$\mathbb{E}_{t,\mathbf{x}_t\sim p_t}\lambda_t \left[\operatorname{div}(\mathbf{s}_\theta)(\mathbf{x}_t, t) + \frac{1}{2}\left\|\mathbf{s}_\theta(\mathbf{x}_t, t)\right\|^2\right] \tag{5}$$

which normally is estimated using sliced score matching/Hutchinson's trace estimator[26, 48]:

$$\mathbb{E}_{t,\epsilon,\mathbf{x}_t\sim p_t}\lambda_t \left[\epsilon^\top D_x\mathbf{s}_\theta(\mathbf{x}_t, t)\epsilon + \frac{1}{2}\left\|\mathbf{s}_\theta(\mathbf{x}_t, t)\right\|^2\right] \tag{6}$$

where ϵ is drawn over some 0 mean and identity covariance distribution like the standard normal distribution or the Rademacher distribution. Unfortunately, the added variance from the divergence computation normally renders this loss unworkable in high dimensions[47], so practitioners instead use the denoising score matching loss[51]

$$\mathbb{E}_{t,\mathbf{x}_0\sim p_0,\mathbf{x}_t\sim p_t(\cdot|\mathbf{x}_0)}\lambda_t \left\|\mathbf{s}_\theta(\mathbf{x}_t,t) - \nabla_x \log p_t(\mathbf{x}_t|\mathbf{x}_0)\right\|^2 \tag{7}$$

where $p_t(\mathbf{x}_t|\mathbf{x}_0)$ is derived from the SDE in Equation 1 and is normally tractable. Once $\mathbf{s}_\theta(\mathbf{x}_t,t)$ is learned, we can construct a generative model by first sampling $\mathbf{x}_T \sim p_T$ and solving the generative SDE from $t=T$ to $t=0$:

$$\mathrm{d}\mathbf{x}_t = (\mathbf{f}(\mathbf{x}_t,t) - g^2(t)\mathbf{s}_\theta(\mathbf{x}_t,t))\mathrm{d}t + g(t)\mathrm{d}\overline{\mathbf{B}}_t \tag{8}$$

Furthermore, there exists a corresponding "probability flow ODE" [49]

$$\mathrm{d}\mathbf{x}_t = (\mathbf{f}(\mathbf{x}_t,t) - \frac{g(t)^2}{2}\nabla_x \log p_t(\mathbf{x}_t))\mathrm{d}t \tag{9}$$

that has the same evolution of p_t as the SDE in Equation 1. This can be approximated using our score network $\mathbf{s}_\theta$ to get a Neural ODE [11]

$$\mathrm{d}\mathbf{x}_t = (\mathbf{f}(\mathbf{x}_t,t) - \frac{g(t)^2}{2}\mathbf{s}_\theta(\mathbf{x}_t,t)\mathrm{d}t \tag{10}$$

which can be used to evaluate exact likelihoods of the data [20].

2.2 Riemannian Diffusion Models

To generalize diffusion models to d-dimensional Riemannian manifolds $\mathcal{M}$, which we assume to be compact, connected, and isometrically embedded in Euclidean space, one adapts the existing machinery to the geometrically more complex space [4]. Riemannian manifolds are deeply analytic constructs, so Euclidean space operations like vector fields $\mathbf{v}$, gradients ∇, and Brownian motion $\mathbf{B}_t$ have natural (in the categorical sense) analogues on $\mathcal{M}$. This allows one to mostly port over the diffusion model machinery from Euclidean space. Here, we highlight some of the core differences.

The forward SDE is the heat equation. The particle dynamics follow a Brownian motion:

$$\mathrm{d}\mathbf{x}_t = \mathrm{d}\mathbf{B}_t \tag{11}$$

which can easily be rescaled by time $\int_0^t g(t)ds$ given a time schedule $g(t)$ (though this is omitted for clarity purposes). Unlike the Euclidean case, here p_t is stationary and approaches the uniform distribution $\mathcal{U}_\mathcal{M}$ as $t \to \infty$ (in practice, this convergence is fast, getting within numerical precision for $t \approx 5$).

The transition density has no closed form. Despite the fact that we work with the most simple SDE, the transition kernel defined by the manifold heat equation $p_t(x_t|x_0)$ has no closed form. This transition kernel is known as the heat kernel, which satisfies Equation 2 with the additional condition that, as $t \to 0$, the kernel approaches δ_{x_0}. We will denote this by $K_\mathcal{M}(x_t|x_0,t)$, and we highlight that, when $\mathcal{M}$ is $\mathbb{R}^d$, this corresponds to a Gaussian and is easy to work with.

This has several major consequences which cause prior to work to favor sliced score matching over denoising score matching. First, to sample a point $x \sim K_\mathcal{M}(\cdot|x_0,t)$, one must simulate a Geodesic Random Walk

$$x_{t+\Delta t} = \exp_x(\sqrt{\Delta t} z) \quad z \sim \mathcal{N}(0, I_d) \in T_x\mathcal{M} \tag{12}$$

where exp is the Riemannian exponential map. Additionally, to calculate $K_\mathcal{M}(\cdot|x_0,t)$ or $\nabla \log K_\mathcal{M}(x|x_0,t)$, one must truncate the eigenfunction representation

$$K_\mathcal{M}^{\mathrm{EF}}(x|x_0,t) = \sum_{i=0}^{\infty} e^{-\lambda_i t} f_i(x_0) f_i(x) \tag{13}$$

Here, f_i, λ_i are the discrete eigenfunctions/eigenvalues of the Laplacian $\Delta f_i = -\lambda_i f_i$ and form an orthonormal basis for all L^2 functions on $\mathcal{M}$. Previous work has also explored the use of the Varadhan approximation for small values of t (which uses the Riemannian logarithmic map) [44]:

$$K_\mathcal{M}(x|x_0,t) \approx \mathcal{N}(0,\sqrt{2t})(\mathrm{dist}_\mathcal{M}(x_0,x)) \implies \nabla_x \log K_\mathcal{M}(x|x_0,t) \approx \frac{1}{2t}\log_x(x_0) \tag{14}$$

3 Method

The key problem with applying denoising score matching in practice is that the heat kernel computation is too expensive and inaccurate. For example, simulating a geodesic random walk is expensive since it requires many exponential maps. Furthermore, the eigenfunction expansion in Equation 13 requires increasingly more eigenfunctions as $t \to 0$ (numbering in the tens of thousands). Worse still, their formulas are not well-known for most manifolds, and, even when explicit formulas exist, they can be numerically unstable (like in the case of S^n). One possible way to alleviate this is to use Varadhan's approximation for small t, but this is also unreliable except for very small t.

To remedy this issue, we instead consider the case of Riemannian symmetric spaces [22]. We emphasize that most manifold generative modeling applications already model on Riemannian Symmetric Spaces like the sphere, torus, or Lie Groups, so we do not lose applicability by restricting our attention here. Furthermore, for surfaces (which have appeared as generative modeling test tasks in the literature [43]), one can always define a generative model by mapping the data points to S^2, learning a generative model there, and mapping back [14]. We empathize that, outside of these two examples, we are unaware of any other manifolds which have been used for Riemannian generative modeling tasks.

3.1 Heat Kernels on Riemannian Symmetric Spaces

In this section, we will define Riemannian Symmetric Spaces and showcase relationships with the heat kernel. We empathize that our exposition is neither rigorous nor fully defines all terms. We urge interested readers to consult a book [22] or monograph [9] for a full treatment of the subject.

Definition 3.1. *A Riemannian Symmetric Space is a Riemannian manifold such that, for all points $x \in \mathcal{M}$, there exists a local isometry s s.t. $s(x) = x$ and $D_x s = -\mathrm{id}_{T_x\mathcal{M}}$.*

This symmetry property is relatively simple but has numerous ramifications. In particular, we can characterize all Riemannian symmetric spaces as a quotient G/K where G is a Lie Group and K is a compact isotropy group.

Examples. *This includes many well known manifolds, such as Lie Groups $G \cong G/\{e\}$ (where $\{e\}$ is the trivial Lie Group), the sphere $S^n \cong SO(n+1)/SO(n)$, and hyperbolic space $H^n \cong SO(n,1)/O(n)$.*

In our paper, we do not consider the case of noncompact Riemannian symmetric spaces, as these are diffeomorphic to Euclidean space. As such, we can reapply the same generative modeling trick that we used for surfaces: map data points to $\mathbb{R}^n$, learn a standard diffusion model there, and map back.

On symmetric spaces, one can define a special structure called the maximal torus which is critical for our analysis. Intuitively, the maximal torus parameterizes the symmetries of the space.

Definition 3.2 (Maximal Torus)**.** *A torus on a Lie group is any compact, connected, and abelian subgroup of G. These are isomorphic to standard tori $T^m \cong (S^1)^m$. A maximal torus T is a torus which is not contained in any other torus. All maximal tori are conjugate. Symmetric spaces inherit maximal tori from their quotient space G.*

Examples. *For the Lie group of unitary matrix $U(n)$, the maximal torus is defined as $T = \{\mathrm{diag}(e^{i\theta_1}, \dots, e^{i\theta_n}) : \theta_k \in [0, 2\pi)\}$. For the sphere S^n, a maximal torus is any great circle.*

We write R^+ as the set of all the positive roots of our symmetric spaces (these are values on the maximal torus), and for a root α we let m_α be the multiplicity (e.g. for spheres, α is 1 and m_α is $d-1$). Furthermore, for a point x, we let h be the "flat" coordinates of x on the maximal torus (e.g. for spheres h is the angle between x and an anchor point x_0). Then, we can rewrite the heat kernel in terms of the maximal torus:

Proposition 3.3 (Heat Kernel Reduces on Maximal Torus)**.** *The Laplace-Beltrami operator on $\mathcal{M}$ (the manifold generalization of the standard Laplacian) induces the "radial" Laplacian on T:*

$$L_r = \Delta_T + \sum_{\alpha \in R^+} m_\alpha \cot(\alpha \cdot h) \frac{\partial}{\partial \alpha} \tag{15}$$

where Δ_T is the standard Laplacian on the torus. As such, the heat kernel reduces to a function of h.

3.2 Improved Heat Kernel Estimation

We now use the fact that the heat kernel is intimately connected with the maximal torus to better estimate the heat kernel values. This greatly improves the speed and fidelity of our numerical evaluation during training.

3.2.1 Eigenfunction Expansion Restricted to the Maximal Torus

We note that the maximal torus relationship in Proposition 3.3 reduces the eigenfunction expansion in Equation 13 to an eigenfunction expansion of the induced Laplacian on the maximal torus. This has implicitly appeared in previous work when defining Riemannian Diffusion Models on S^2 and $SO(3)$ [4, 34], allowing one to rewrite the summation as, respectively

$$K_{S^2}(x|x_0,t) = \frac{1}{4\pi}\sum_{l=0}^{\infty}(2l+1)P_l(\langle x, x_0\rangle)e^{-l(l+1)t} \tag{16}$$

where P_l are the Legendre Polynomials and

$$K_{SO(3)}(x|x_0,t) = \frac{1}{8\pi^2}\sum_{l=0}^{\infty} e^{-2l(l+1)t}\frac{\sin(2l+1)\theta/2}{\sin(\theta/2)} \quad \theta \text{ is the angle between } x, x_0 \tag{17}$$

By making this relationship explicit, we can draw upon similar formulas for other symmetric spaces, e.g. the hypersphere [39]

$$K_{S^n}(x|x_0,t) = \frac{1}{V(S^n)}\sum_{l=0}^{\infty}\frac{2l+n-1}{n-1}G_l^{(n-1)/2}(\langle x, x_0\rangle)e^{-l(l+d-1)t} \tag{18}$$

where G_l^α are the Gegenbauer polynomials and $V(S^n)$ is the volume of S^n. The new summation works by effectively "collapsing" the summation for eigenfunctions f_i with the same eigenvalue λ_i. This drastically reduces the number of computations required from $O(M^{\dim \mathcal{M}})$ to $O(M^{\dim T})$, where M is the cutoff value. Furthermore, this also tends to greatly simplify the explicit formula. As an example, for S^n this reduces the computation from $O(M^n)$ to $O(M)$ (since we no longer need to evaluate $O(i)$ eigenfunctions for each eigenvalue λ_i) and avoids the computation of numerically unstable hyperspherical harmonics.

3.2.2 Controlling Small Time Errors

While the torus eigenfunction representation greatly reduces the computational cost (particularly for several higher-dimensional manifolds), they still require thousands of eigenfunctions for small values of t. Worse still, numerical error persists: for small values of t, computing the eigenfunction expansion can easily cause overflow errors that even double precision can't resolve. To this end, we examine several refined versions of the Varadhan approximation that use the fact that the manifold is a Riemannian symmetric space. These approximations can allow us to control the number of eigenfunctions required and, in some cases, completely obviate the need for them altogether.

The Schwinger-Dewitt Approximation

The Varadhan approximation is built by approximating the heat kernel with a Gaussian distribution with respect to the Riemannian distance function. However, doing this does not account for the curvature of the manifold. By accounting for this curvature, we derive the Schwinger-Dewitt approximation [9]:

$$K_{\mathcal{M}}^{\text{SD}}(x|x_0,t) = \frac{\overline{\Delta}_{x_0}(x)^{1/2}e^{-\frac{d_{\mathcal{M}}(x_0,x)^2}{4t}+\frac{tR}{6}}}{(4\pi t)^{\dim\mathcal{M}/2}} \tag{19}$$

Here $\overline{\Delta}_{x_0}(x) = \det(D_{x_0}\exp_{x_0}(\log_{x_0}(x)))$ is the (unnormalized) change of volume term introduced by the exponential map, and R is the scalar curvature of the manifold. Generally, this is much more stable than Varadhan's approximation as it better accounts for the curvature of the manifold, retaining accuracy up to moderate time values.

Δ appears to be a rather computationally demanding term. Indeed, naïve calculations require the formation of the full Jacobian matrix and a determinant computation, which scales poorly with dimensions and is completely inaccessible in higher dimensions. However, we again emphasize the fact that we are working with symmetric spaces; here, Δ has a particularly simple formula defined by our flat coordinate h from above:

$$\overline{\Delta}_{x_0}(x) = \prod_{\alpha \in R^+} \left(\frac{\alpha \cdot h}{\sin(\alpha \cdot h)} \right)^{m_\alpha} \tag{20}$$

Sum Over Paths

The fact that we can derive a better approximation using a different power of Δ points to deeper connections between the heat kernel and the "Gaussian" with respect to distance. We draw inspiration from several Euclidean case examples, such as the flat torus [28] or the unit interval [36]. For these cases, the heat kernel is derived by summing a Gaussian over all possible paths connecting x_0 and x. While this formula does not exactly lift over to Riemannian symmetric spaces, there exists a facsimile for Lie Groups [9]:

$$K_{\mathcal{M}}^{\mathrm{SOP}}(x|x_0, t) = \frac{e^{t\rho^2}}{(4\pi t)^{\dim \mathcal{M}/2}} \sum_{2\pi n \in \Gamma} \prod_{\alpha \in R^+} \left(\frac{\alpha \cdot (h + 2\pi n)}{2 \sin(\alpha \cdot h/2)} \right) e^{-\frac{(h+2\pi n)^2}{4t}} \tag{21}$$

Here, Γ is the set of all points in the tangent space to the identity which exponentiate back to the source point (e.g. in spheres this is all distances which integral multiples of 2π), and ρ^2 is a manifold specific constant. We note that the product over R^+ is exactly the Δ change in variables above since $m_\alpha = 1$, but we simply extend this to every other root.

Generally, this formula is rather powerful as it gives us an exact (albeit infinite) representation for the heat kernel. Compared to the eigenfunction expansion in Equation 13, the Sum Over Paths representation is accurate for small t, which nicely complements the fact that the eigenfunction representation is accurate for large t. This formula does generalize to split-rank Riemannian symmetric spaces like odd dimensional spheres. However, we did not pursue these formulas further since the formulas are much more complex due to the appearance of intertwining operators.

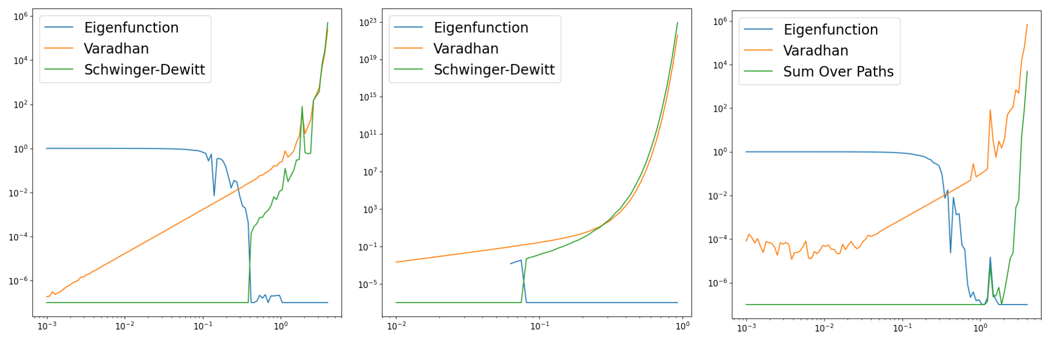
3.2.3 A Unified Heat Kernel Estimator

We unify these approximants into a single heat kernel estimator. Our computation method splits up the heat kernel evaluation based on time steps, and applies an eigenfunction summation or an improved small time approximation accordingly. This allows us to effectively control the errors at both the small and large time steps while significantly reducing the number of function evaluations. Our full algorithm is outlined in Algorithm 1.

Algorithm 1: Heat Kernel Computation

```
Hyperparameters: Riemannian symmetric space $\mathcal{M}$, number of eigenfunctions $n_e$, time value
  cutoff $\tau$, (optional, depending on if $\mathcal{M}$ is a Lie Group) number of paths $n_p$
Input: source $x_0$, time $t$, query value $x$
Compute
if $t < \tau$ then
    if $\mathcal{M}$ is a Lie Group then
        return $K_{\mathcal{M}}^{\mathrm{SOP}}(x|x_0, t)$ truncated to $|n| < n_p$ in the summation over $\Gamma$.
    else
        return $K_{\mathcal{M}}^{\mathrm{SD}}(x|x_0, t)$.
    end
else
    return $K_{\mathcal{M}}^{\mathrm{EF}}(x|x_0, t)$ truncated to $|n| < n_e$.
end
```

We can compute $\nabla_x \log K_{\mathcal{M}}$ using conventional autodifferentiation tools. As this is the score quantity used for training, we ablate the accuracy of our various heat kernel approximations in Figure 1 for S^2, S^{127}, and $SO(3)$. In general, we found that computing with standard eigenfunctions was too costly and too prone to numerical blowup, and Varadhan's approximation was simply too inaccurate. In particular, for S^{127}, preexisting methods would not work since $K_{\mathcal{M}}^{\mathrm{EF}}$ NaNs out before Varadhan becomes accurate.



(a) S^2 (b) S^{127} (c) $SO(3)$

Figure 1: **We compare the various heat kernel estimators on a variety of manifolds.** We plot the relative error compared with t. Our improved small time asymptotics allow us to control the numerical error, while baseline Varadhan is insufficient. For S^{127}, Varadhan will still produce an error of 10% when the eigenfunction expansion NaNs out, so we need to use Schwinger-Dewitt

3.3 Exact Heat Kernel Sampling

To train with the denoising score matching objective, one must produce heat kernel samples to optimize with. Typically, Riemannian Diffusion Models sample by discretizing a Brownian motion through a geodesic random walk. However, this can be slow as it requires taking many computationally expensive exponential map steps on $\mathcal{M}$ and can drift off the manifold due to compounding numerical error. In this section, we discuss strategies to sample from the heat kernel quickly and exactly.

Cheap Rejection Sampling Methods

Using our cheap heat kernel evaluations, we can sample using rejection sampling. The key detail is the prior distribution, which needs to be cheap to evaluate, easy to sample from, and must not deviate from the heat kernel too much. For large time steps t the natural prior distribution is uniform. Conversely, for small time steps, we instead use the wrapped Gaussian distribution, which can be sampled by passing a tangent space Gaussian through the exponential map and has a density

$$p_{\text{wrap}}(x|x_0, t) = \frac{1}{(4\pi t)^{\dim \mathcal{M}/2}} \sum_{2\pi n \in \Gamma} \prod_{\alpha \in R^+} \left(\frac{\sin(\alpha \cdot h)}{\alpha \cdot (h + 2\pi n)} \right) e^{-\frac{(h+2\pi n)^2}{4t}} \tag{22}$$

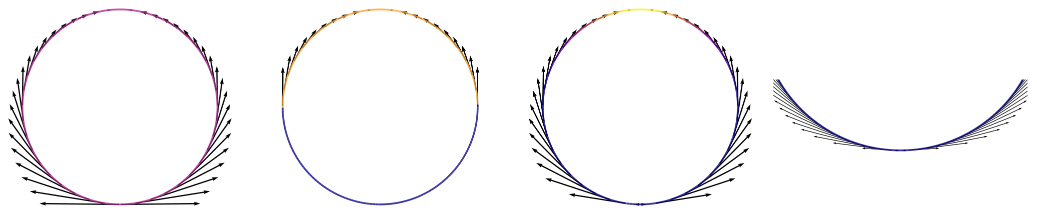
Heat Kernel ODE Sampling

We notice that we can apply the probability flow ODE to sample from the heat kernel exactly. In particular, we draw a sample $x_T \sim \mathcal{U}_{\mathcal{M}}$, where T is large enough s.t. $K_{\mathcal{M}}(\cdot|x_0, T)$ is (numerically) uniform, and solve the ODE $\frac{d}{ds} x_T = -\frac{1}{2} \nabla_x \log K_{\mathcal{M}}(x_s|x_0, s)$ from $s = T$ to $s = t$. By the same construction as the probability flow ODE, this is guaranteed to produce samples from $K_{\mathcal{M}}(\cdot|x_0, t)$.

This is solvable as a manifold ODE, as previous works have already developed adaptive manifold ODE solvers [37]. Furthermore, by Proposition 3.3, we can restrict our vector field to the maximal torus and solve it there. Note that this allows us to use preexisting Euclidean space solvers since the torus is effectively Euclidean space. Lastly, we can scale the time schedule of the ODE (with a scheme like a variance-exploding schedule) to stabilize the numerical values.

4 Related Work

Our work exists in the established framework of differential equation-based Riemannian generative models. Early methods generalized Neural ODEs to manifolds[15, 37, 38], enabling training with maximal likelihood. More recent methods attempt to remove the simulation components[3, 43], but this results in unscalable or biased objectives. We instead work with diffusion models, which are based on scores and SDEs and do not have any of these issues. In particular, we aim to resolve the main gap that prevents Riemannian Diffusion Models from scaling to high dimensions.



(a) $t = 0$ FM Path. (b) $t = 0.5$ FM Path. (c) Diffusion SM path. (d) Closeup of Figure 2c

Figure 2: **We visualize the vector fields generated by the flow matching geodesic path and our score matching diffusion path.** These are done on S^1. (a) The flow matching path has a discontinuity at the pole. (b) The marginal densities of the flow matching path are not smooth and transition sharply at the boundary. (c) Our score matching path has a smooth density and smooth vectors. (d) At the pole, our score matching path anneals to 0 to maintain continuity.

Method	Volcano	Earthquake	Flood	Fire
Sliced Score Matching	-4.92 $\pm$ 0.25	-0.19 $\pm$ 0.07	0.45 $\pm$ 0.17	-1.33 $\pm$ 0.06
Denoising Score Matching (inaccurate)	-1.28 $\pm$ 0.28	0.13 $\pm$ 0.03	0.73 $\pm$ 0.04	-0.60 $\pm$ 0.18
Denoising Score Matching (accurate)	-4.69 $\pm$ 0.29	-0.27 $\pm$ 0.05	0.44 $\pm$ 0.03	-1.51 $\pm$ 0.13

Table 1: **We measure the improvement of our improved heat kernel estimator on downstream climate science tasks and report negative log likelihood (↓).** Without our accurate heat kernel estimator, the denoising score matching loss produces substandard results.

Riemannian Flow Matching (RFM) [10] is a very recent work that attempts to achieve similar goals (i.e. scaling to high dimensions) by generalizing flow matching to Riemannian manifolds [35]. The fundamental difficulty is that one must design smooth vector fields that flows from a base distribution (ie the uniform distribution) to a data point. RFM introduces several geodesic-based vector fields, but these break the smoothness assumption and the theoretical framework (see Figure 2). We found that, although RFM is able to easily learn relatively simple distributions, this non-smoothness is highly detrimental for more nontrivial densities and can be crippling in high dimensions (see Figure 3). We also note that, similar to the Euclidean case, RFM with the diffusion path corresponds to score matching with the probability flow ODE, so our work provides a computation for this path.

5 Experiments

5.1 Simple Test Tasks

We start by comparing our accurate denoising score matching objective with the inaccurate version suggested by [4] based on 50 eigenfunctions. We test on the compiled Earth science datasets from [38], detailing results are in Table 1. Generally, our accurate heat kernel results in a substantial improvement and matches sliced score matching. Note that we do not expect our method to outperform sliced score matching since these datasets are low dimensional.

We also compare directly with RFMs on a series of increasingly complex checkerboard datasets on the flat torus. These datasets have appeared in prior work to measure model quality [3, 5, 37]. As a result of the non-smooth vector field dynamics, we find that RFMs degrade in performance as the checkerboard increases in complexity, and is unable to learn past a certain point. Our visualized results are given in Figure 3.

5.2 Learning the Wilson Action on $SU(3)$ Lattices.

We apply our method to learn $SU(3)$ configurations on a 4×4 lattice. In particular, we generate data on $SU(3)^{4\times 4}$ according to the Wilson Action [53] $p(K) \propto e^{-S(K)}$, where $S(K)$ is defined as:

$$S(K) = -\frac{\beta}{3} \sum_{x,y \in \mathbb{Z}_4^2} \mathrm{Re}\,\mathrm{tr}(K_{x,y} K_{x+1,y} K^*_{x+1,y+1} K^*_{x,y+1}) \tag{23}$$

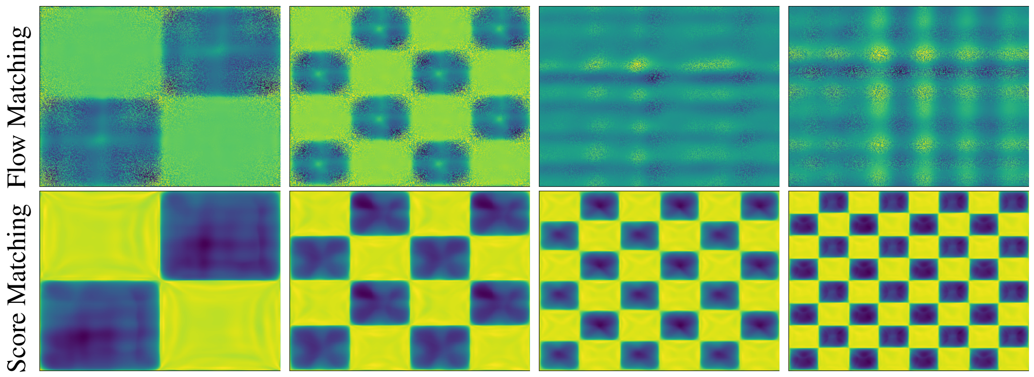


Figure 3: **We compare Riemannian score matching and flow matching on increasingly complex checkerboard patterns on the torus.** On simple checkerboards, Flow matching learns suboptimal distributions with noticeable artifacts like blurriness and spurious peaks, and fails for more complex checkerboards. Conversely, Riemannian score matching/diffusion learns accurate densities.

Method	SVHN	Places365	LSUN	iSUN	Texture
SSD+[45]	31.19	77.74	79.39	80.85	66.63
KNN+[50]	39.23	80.74	48.99	74.99	57.15
CIDER (penultimate layer)	23.09	79.63	16.16	71.68	43.87
CIDER (hypersphere)	93.53	89.92	93.68	89.92	92.41
CIDER (hypersphere) + Diffusion	28.95	76.54	35.78	74.17	62.87

Table 2: **We compare contrastive learning OOD detection methods on CIFAR-100.** We report false positive rates (↓) for 0.05 false negative rate. The hyperspherical embeddings produce very bad results, but with a Riemannian Diffusion Model, it is competitive with or surpass the state of the art.

We take $\beta = 9$ for our purposes. Our model trains stably and can learn the density, achieving an ESS of 0.62. This is not the standard variational inference training procedure [6], since it requires samples to train with, but concurrent work has shown that this type of training can be coupled with diffusion guidance to improve variational inference methods [17]. As such, our model has the potential to improve $SU(n)$ lattice QCD samplers, although we leave this task for future work. We also note that our model can likely be further improved by building data symmetries into the score network [30].

5.3 Contrastively Learned Hyperspherical Embeddings

Finally, we examine contrastively learning [12]. Standard contrastive losses optimize embeddings of the data that lie on the hypersphere. Paradoxically, these embeddings are unsuitable for most downstream tasks, so practitioners instead use the penultimate layer [2]. This degrades interpretability, since the theoretical analyses in this field work on the hyperspherical embeddings [52].

We investigate this issue further in the context of out of distribution (OOD) detection. We use the pretrained embedding network from CIDER[40]. Using the hyperspherical representation for OOD detection produces very bad results. However, using likelihoods from our Riemannian Diffusion Model stabilizes performance and achieves comparable results with other penultimate feature-based methods (see Figure 2). We emphasize that the embedding network has been tuned to optimize the performance using the penultimate layer. Since our established theory exclusively focuses on the properties of the hyperspherical embedding, the fact that our Riemannian Diffusion Models can extract a comparable representation can lead to more principled improvements for future work.

6 Conclusion

We have introduced several practical improvements for Riemannian Diffusion Models that leverage the fact that most relevant manifolds are Riemannian symmetric spaces. Our improved capabilities allow us, for the first time, to scale differential equation manifold models to hundreds of dimensions, where we showcase applications in lattice QCD and constrastive learning. We hope that our improvements help open the door to the broader adoption of Riemannian generative modeling techniques.

References

[1] Belal E. Baaquie. Path-integral derivation of the heat kernel on $\mathrm{SU}(n)$ space. *Phys. Rev. D*, 32:1007–1010, Aug 1985. doi: 10.1103/PhysRevD.32.1007. URL `https://link.aps.org/doi/10.1103/PhysRevD.32.1007`.

[2] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari S. Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Grégoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning. *ArXiv*, abs/2304.12210, 2023.

[3] Heli Ben-Hamu, Samuel Cohen, Joey Bose, Brandon Amos, Maximilian Nickel, Aditya Grover, Ricky T. Q. Chen, and Yaron Lipman. Matching normalizing flows and probability paths on manifolds. In *International Conference on Machine Learning*, 2022.

[4] Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and A. Doucet. Riemannian score-based generative modeling. *ArXiv*, abs/2202.02763, 2022.

[5] A. Bose, Ariella Smofsky, Renjie Liao, P. Panangaden, and William L. Hamilton. Latent variable modelling with hyperbolic normalizing flows. In *International Conference on Machine Learning*, 2020.

[6] Denis Boyda, Gurtej Kanwar, Sébastien Racanière, Danilo Jimenez Rezende, Michael S Albergo, Kyle Cranmer, Daniel C. Hackett, and Phiala E. Shanahan. Sampling using su(n) gauge equivariant flows. *ArXiv*, abs/2008.05456, 2020.

[7] Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. *ArXiv*, abs/2003.13913, 2020.

[8] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur D. Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34:18–42, 2016.

[9] Roberto Camporesi. Harmonic analysis and propagators on homogeneous spaces. *Physics Reports*, 196: 1–134, 1990.

[10] Ricky T. Q. Chen and Yaron Lipman. Riemannian flow matching on general geometries. *ArXiv*, abs/2302.03660, 2023.

[11] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Kristjanson Duvenaud. Neural ordinary differential equations. In *Neural Information Processing Systems*, 2018.

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *ArXiv*, abs/2002.05709, 2020.

[13] Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and T. Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. *ArXiv*, abs/2210.01776, 2022.

[14] Victor D. Dorobantu, Charlotte Borcherds, and Yisong Yue. Conformal generative modeling on triangulated surfaces. *ArXiv*, abs/2303.10251, 2023.

[15] Luca Falorsi and Patrick Forr'e. Neural ordinary differential equations on manifolds. *ArXiv*, abs/2006.06663, 2020.

[16] H. D. Fegan. The heat equation on a compact lie group. *Transactions of the American Mathematical Society*, 246:339–357, 1978. ISSN 00029947. URL `http://www.jstor.org/stable/1997977`.

[17] Tomas Geffner, George Papamakarios, and Andriy Mnih. Score modeling for simulation-based inference. *ArXiv*, abs/2209.14249, 2022.

[18] Mevlana Gemici, Danilo Jimenez Rezende, and Shakir Mohamed. Normalizing flows on riemannian manifolds. *ArXiv*, abs/1611.02304, 2016.

[19] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[20] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Kristjanson Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *ArXiv*, abs/1810.01367, 2018.

[21] Alexander Grigor'yan. Spectral theory and geometry: Estimates of heat kernels on riemannian manifolds. 1999.

[22] Sigurdur Helgason. Differential geometry, lie groups, and symmetric spaces. 1978.

[23] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv: Learning*, 2016.

[24] Jonathan Ho, Ajay Jain, and P. Abbeel. Denoising diffusion probabilistic models. *ArXiv*, abs/2006.11239, 2020.

[25] Chin-Wei Huang, Milad Aghajohari, A. Bose, P. Panangaden, and Aaron C. Courville. Riemannian diffusion models. *ArXiv*, abs/2208.07949, 2022.

[26] Michael F. Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics - Simulation and Computation*, 18:1059–1076, 1989.

[27] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *J. Mach. Learn. Res.*, 6:695–709, 2005.

[28] Bowen Jing, Gabriele Corso, Jeffrey Chang, Regina Barzilay, and T. Jaakkola. Torsional diffusion for molecular conformer generation. *ArXiv*, abs/2206.01729, 2022.

[29] Jurgen Jost. Riemannian geometry and geometric analysis. 1995.

[30] Isay Katsman, Aaron Lou, Derek Lim, Qingxuan Jiang, Ser-Nam Lim, and Christopher De Sa. Equivariant manifold flows. In *Neural Information Processing Systems*, 2021.

[31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[32] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *ArXiv*, abs/2107.00630, 2021.

[33] Peter E. Kloeden and Eckhard Platen. The numerical solution of stochastic differential equations. *The Journal of the Australian Mathematical Society. Series B. Applied Mathematics*, 20:8 – 12, 1977.

[34] Adam Leach, Sebastian M Schmon, Matteo T. Degiacomi, and Chris G. Willcocks. Denoising diffusion probabilistic models on SO(3) for rotational alignment. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022. URL `https://openreview.net/forum?id=BY88eBbkpe5`.

[35] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *ArXiv*, abs/2210.02747, 2022.

[36] Aaron Lou and Stefano Ermon. Reflected diffusion models. *ArXiv*, abs/2304.04740, 2023.

[37] Aaron Lou, Derek Lim, Isay Katsman, Leo Huang, Qingxuan Jiang, Ser-Nam Lim, and Christopher De Sa. Neural manifold ordinary differential equations. *ArXiv*, abs/2006.10254, 2020.

[38] Emile Mathieu and Maximilian Nickel. Riemannian continuous normalizing flows. *ArXiv*, abs/2006.10605, 2020.

[39] Aleksandar Mijatović, Veno Mramor, and Gerónimo Uribe Bravo. An algorithm for simulating brownian increments on a sphere. *Journal of Physics A: Mathematical and Theoretical*, 54(11):115205, feb 2021. doi: 10.1088/1751-8121/abd69f. URL `https://dx.doi.org/10.1088/1751-8121/abd69f`.

[40] Yifei Ming, Yiyou Sun, Ousmane Dia, and Yixuan Li. How to exploit hyperspherical embeddings for out-of-distribution detection? In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=aEFaE0W5pAd`.

[41] Boris Polyak and Anatoli B. Juditsky. Acceleration of stochastic approximation by averaging. *Siam Journal on Control and Optimization*, 30:838–855, 1992.

[42] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[43] Noam Rozen, Aditya Grover, Maximilian Nickel, and Yaron Lipman. Moser flow: Divergence-based generative modeling on manifolds. In *Neural Information Processing Systems*, 2021.

[44] Laurent Salo-Coste. The heat kernel and its estimates. 2009.

[45] Vikash Sehwag, Mung Chiang, and Prateek Mittal. Ssd: A unified framework for self-supervised outlier detection. *ArXiv*, abs/2103.12051, 2021.

[46] Jascha Narain Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *ArXiv*, abs/1503.03585, 2015.

[47] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *ArXiv*, abs/1907.05600, 2019.

[48] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Conference on Uncertainty in Artificial Intelligence*, 2019.

[49] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=PxTIG12RRHS`.

[50] Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. In *International Conference on Machine Learning*, 2022.

[51] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23:1661–1674, 2011.

[52] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. *ArXiv*, abs/2005.10242, 2020.

[53] Kenneth G. Wilson. Confinement of quarks. *Physical Review D*, 10:2445–2459, 1974.

A Explicit Heat Kernel Formulas

In this section, we highlight the formulas we used for computing the heat kernel.

Torus. The Torus $T^n \cong (S^1)^n$ can be realized as a flat torus $[0, 2\pi)^n$, where each coordinate represents the angular component. Under this construction, we can compute the kernel for each coordinate in $S^1 \cong [0, 2\pi)$ and then take the product. The eigenfunction expansion is

$$K_{S^1}(y|x,t) = \frac{1}{\pi}\left(\frac{1}{2} + \sum_{k=1}^{\infty} e^{-k^2 t}(\cos(kx)\cos(ky) + \sin(kx)\sin(ky)\right) \tag{24}$$

The heat kernel also admits a sum over paths representation. In particular, this agrees with the wrapped probability since the change of volume term is 1:

$$K_{S^1}(y|x,t) = \frac{1}{\sqrt{4\pi t}} \sum_{k=-\infty}^{\infty} e^{-\frac{(y-x+2\pi k)^2}{4t}} \tag{25}$$

Spheres. The sphere S^n is the set $\{x \in \mathbb{R}^{n+1} : \|x\| = 1\}$. We use the formulas (eigenfunction and Schwinger-Dewitt) given in the main text, noting that the maximal torus value can be extracted with the equation $\theta = \arccos(x \cdot y)$ (ie the geodesic distance).

$\mathbf{SO(3)}$. $SO(3)$ is the Lie Group $\{X \in \mathbb{R}^{3\times 3} : XX^\top = I, \det(X) = 1\}$. We have already given the eigenfunction expansion in the main text, but the sum over paths method can be derived from the fact that the maximal torus value θ is the distance between x and y on the manifold.

$\mathbf{SU(3)}$. $SU(3)$ is the (real) Lie Group $\{X \in \mathbb{C}^{3\times 3} : XX^H = I, \det(X) = 1\}$. The eigenfunction expansion can be derived from the character classes [16], and the sum over paths representation can be derived directly [1].

B Experimental Details

B.1 Heat Kernel Estimates

We sample a random point (based off of the heat kernel probability) for each time step to compute.

$\mathbf{S^2}$. We use 10000 eigenfunctions for the ground truth and 10 for our comparison.

$\mathbf{S^{127}}$. We use 50000 eigenfunctions evaluated at double precision for our ground truth and 100 for our comparison.

$\mathbf{SO(3)}$. We use 10000 eigenfunctions for our ground truth and 50 for our comparison. We sum over 10 paths.

B.2 Earth Science Datasets

We do not perform a full hyperparameter search. We use a very similar architecture to the one used in Bortoli et al. [4] except we use the SiLU activation function without a learnable parameter [23] and a learning rate of $5e-4$.

B.3 2D Torus

We use a standard MLP with 4 hidden layers and the SiLU activation function and learn with the Adam optimizer with learning rate set to $1e-3$ [31]. However, we transform the input $x \to \sin(kx), \cos(kx)$ where k ranges from 1 to 6. This was done to ensure that the input respects the boundary constraints. We note that this architecture is generally quite powerful, as the Fourier coefficients can capture finer grain features, but this was optimized for the flow matching baseline. In particular, score matching works with significantly fewer Fourier coefficients. We train for 100000 gradient updates with a batch size of 100 (each batch is randomly sampled from the checkerboard).

B.4 SU(3) Lattice

We generate our 20000 ground truth samples using Riemannian Langevin dynamics with a step size of $1e-3$ for 10000 update iterations. Our model is similar to the model used in Kingma et al. [32], except we circular pad the convnet and use 3 layers for each up-down block instead. We input a compressed version of the 3×3 $SU(n)$ matrix, making the input size 18. We train with a learning rate of $5e-4$ and perform 1000000 updates with a batch size of 512. To evaluate, we use an 0.999 exponential moving average [41] and sample using the manifold ODE sampler [37].

B.5 Contrastive Learning

We use the pretrained networks given by Ming et al. [40] to construct our hyperspherical embeddings. Our Riemannian diffusion model is similar to the simplex diffusion model given by [36], although we use 3 layers instead of 4. We train using the Adam optimizer with a learning rate of $5e-4$, performing a 0.999 EMA before using the manifold ODE solver to evaluate likelihoods.