Data-Driven Riemannian Geometry for Generative and Regression Models on Manifolds

Yizhu Wang

Submitted in fulfilment of the requirements for the
Degree of Doctor of Philosophy

School of Mathematics and Statistics
College of Science and Engineering
University of Glasgow

May 2026

Abstract

Many real-world datasets are high-dimensional in their ambient representation but exhibit an intrinsic low-dimensional structure induced by an unknown manifold. When such geometric structure is ignored, probabilistic models built on Euclidean assumptions may induce misleading notions of similarity, produce unreliable uncertainty estimates, and perform poorly in both generative and regression tasks. This thesis develops geometry-aware probabilistic learning methods that explicitly account for intrinsic manifold structure inferred from data.

The first part of the thesis addresses generative modelling on unknown manifolds through a geometry-aware latent diffusion framework. An Intrinsic Hybrid Latent Diffusion Model (ILDM) is introduced, in which the latent space is interpreted as a chart of an unknown manifold endowed with a probabilistic geometry induced by a pretrained decoder. Unlike standard latent diffusion models that impose Euclidean dynamics in the latent space, ILDM defines a hybrid diffusion process that adapts to geometric uncertainty. Riemannian Brownian motion is used in regions where the latent geometry is well supported by data, while Euclidean dynamics are employed in areas of high uncertainty. A corresponding score-based model is learned from simulated hybrid trajectories using an approximate denoising objective. The reverse-time process combines Riemannian and Euclidean Langevin dynamics to generate samples that better respect the underlying manifold structure. Experimental results on image and medical datasets demonstrate improved generation quality compared to conventional diffusion and latent diffusion models.

The second part of the thesis focuses on regression on unknown manifolds through intrinsic Gaussian process models. A probabilistic latent representation of the data manifold is learned using a probabilistic latent manifold model, yielding both a latent coordinate system and a distribution over Riemannian metrics. The learned metric, together with uncertainty information from the latent mapping, is used to characterise the geometry and boundary of the manifold. Brownian motion is simulated on the inferred manifold using the probabilistic metric, and the corresponding heat kernel—approximated via transition densities—is employed as the covariance function of an intrinsic Gaussian process. The resulting model, referred to as Gaussian Processes on Unknown Manifolds (GPUM),

enables regression that respects intrinsic geometry without requiring explicit geodesic distance computations. Its performance is demonstrated on synthetic data and real-world datasets, including point clouds, WiFi signal measurements, and image data, and is compared against Euclidean and graph-based Gaussian process baselines.

Together, these contributions provide a probabilistic framework for learning, inference, and generation on unknown manifolds, highlighting the importance of incorporating geometry and uncertainty into probabilistic models.

Contents

List of Tables

List of Figures

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisors, Dr Mu Niu and Dr Xiaochen Yang, for their guidance, patience, and continuous support throughout my doctoral studies. Their careful supervision, thoughtful advice, and high academic standards have shaped both my research and my approach to academic work. I am particularly grateful for our many discussions, which helped clarify and resolve a number of difficult problems.

I am grateful to the School of Mathematics and Statistics at the University of Glasgow for providing a supportive and stimulating academic environment. The resources, facilities, and collegial atmosphere within the School made my years of study both productive and enjoyable. In particular, I would like to thank the School for the conference travel funding, which enabled me to attend academic conferences, engage with researchers from different fields, and gradually refine my research interests and direction. These opportunities played an important role in the development of this dissertation.

My heartfelt thanks go to my parents and family for their unconditional love, understanding, and encouragement. Their continued support gave me the strength to persist through challenges and maintain motivation throughout this long journey.

I would like to thank my colleagues and fellow students for their companionship and support during my PhD. As many of us were studying abroad, we shared not only academic discussions but also many important moments away from home, which made this journey far less lonely. I am also grateful to my friends back in my hometown, whose constant encouragement and understanding continued to support me through difficult times.

Declaration

I declare that all the work presented in this thesis has been carried out by myself under the supervision of Dr Mu Niu and Dr Xiaochen Yang, except where explicitly stated otherwise. This thesis represents work completed between 2022 and 2024 in the School of Mathematics and Statistics at the University of Glasgow in fulfilment of the requirements for the degree of Doctor of Philosophy.

Parts of this thesis are based on material that is currently under peer review. In particular, the work presented in Chapters 3–5 draws upon the following publications:

- Yizhu Wang, Mu Niu, and Xiaocheng Yang. "Intrinsic-Hybrid Latent Diffusion Models for Generative Modelling on Unknown Manifolds." Manuscript under review at *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

In this work, I am the first author and made the primary contributions. Specifically, I developed the main research ideas and methodology, implemented the proposed approach, carried out the experiments, and performed the empirical analysis. I also led the writing of the manuscript, with feedback and revisions from the co-authors.

Parts of this thesis are based on material that has been previously published. In particular, the work presented in Chapter 2 and Chapter 6 is based on the following publication:

- Mu Niu, Zhenwen Dai, Pokman Cheung, and Yizhu Wang. "Intrinsic Gaussian Process on Unknown Manifolds with Probabilistic Metrics." *Journal of Machine Learning Research*, vol. 24, no. 104, pp. 1–42, 2023. https://jmlr.org/papers/v24/22-0627.html.

In this work, I am the last author and contributed to the comparative evaluation and parts of the experimental study. I also contributed to discussions on the development of the work and to the revision of the manuscript.

Notation

Symbol	Description
$\mathcal{M}$	Smooth Riemannian manifold
d	Intrinsic dimension of $\mathcal{M}$
$x \in \mathcal{M}$	A point on the manifold
$T_x\mathcal{M}$	Tangent space at x
g_x	Riemannian metric at x
$g(x)$	Metric tensor in local coordinates
ϕ	Mapping from latent space to manifold
$J_\phi(x)$	Jacobian of ϕ at x
Δ	Laplace–Beltrami operator
$K_{\text{heat}}(s_0, s, t)$	Heat kernel
B_t	Brownian motion on $\mathcal{M}$
$\mathbf{x}$	Observed data
$\mathbf{z}$	Latent variable
$p(\mathbf{x})$	Data distribution
$q(\mathbf{z}\|\mathbf{x})$	Variational posterior
$p(\mathbf{x}\|\mathbf{z})$	Likelihood / decoder
$\nabla_x \log p(x)$	Score function
σ_t	Noise scale at time t
$\mathbb{E}[\cdot]$	Expectation operator
$\|\cdot\|$	Euclidean norm

Abbreviations

Abbreviation	Meaning
ILDM	Intrinsic Hybrid Latent Diffusion Model
LDM	Latent Diffusion Model
DM	Diffusion Model
VAE	Variational Autoencoder
GPLVM	Gaussian Process Latent Variable Model
GP	Gaussian Process
GPUM	Gaussian Processes on Unknown Manifolds
BM	Brownian Motion
SDE	Stochastic Differential Equation
FID	Fréchet Inception Distance
LPIPS	Learned Perceptual Image Patch Similarity
KDE	Kernel Density Estimation
ARD	Automatic Relevance Determination
CNN	Convolutional Neural Network

Chapter 1

Introduction

Modern machine learning models are increasingly applied to data that are high-dimensional in their ambient representation but intrinsically governed by low-dimensional geometric structure.

Such data arise in a wide range of applications, including images, medical data, and scientific simulations. For example, natural images are typically represented as high-dimensional pixel arrays, yet their variability is largely governed by a small number of underlying factors such as object shape, pose, and illumination; similarly, human motion data and speech signals are known to evolve on low-dimensional manifolds embedded in high-dimensional spaces.

Although observations are embedded in high-dimensional Euclidean spaces, empirical and theoretical studies suggest that their probability mass concentrates near unknown low-dimensional nonlinear manifolds [25, 62]. This statistical regularity, commonly referred to as the manifold hypothesis, implies that the effective support of complex data distributions can be well approximated by low-dimensional geometric structures embedded in the ambient space [24, 25, 62]. As a result, a large portion of the apparent high dimensionality observed in real-world datasets is redundant, and their intrinsic variability can often be captured by substantially lower-dimensional representations.

The geometry of this underlying manifold plays a fundamental role in shaping meaningful notions of similarity, variability, and connectivity among data points. Distances measured along the manifold capture intrinsic relationships that are not reflected by ambient Euclidean distance alone, particularly when the manifold is curved, folded, or non-uniformly samplled [24, 25, 64]. Consequently, models that fail to account for such geometric structure may suffer from distorted representations and misleading notions of proximity, even when large amounts of data are available. This motivates the development of generative and representation learning frameworks that explicitly incorporate intrinsic

geometric structure into their modeling assumptions.

This thesis investigates probabilistic learning methods that explicitly incorporate manifold geometry. By combining ideas from differential geometry, stochastic processes, and modern probabilistic modeling such as Gaussian Process Latent Variable Model (GPLVM), variational autoencoders and diffusion-based generative models, the work develops geometry-aware frameworks for both regression and generative modeling. A central theme throughout the thesis is that geometric structure should not be treated as an implicit byproduct of manifold learning, but rather as an important object that is inferred from data and directly incorporated into probabilistic inference and stochastic dynamics.

This chapter introduces the thesis and provides an overview of its scope and structure. It begins by discussing the motivation for the research, followed by a statement of the main goals pursued in this work. The chapter concludes with an outline of the organisation of the remainder of the thesis.

1.1 Motivation

A wide range of modern datasets are high-dimensional in their ambient representation, yet their essential variability is governed by unknown low-dimensional structure embedded within the observation space. Such data arise in many domains, including images, medical measurements, and scientific simulations, where observations concentrate around an unknown manifold. This perspective, commonly referred to as the manifold hypothesis, has been widely supported in both theory and practice [25]. A simple illustration is shown in Figure 1.1, where data embedded in a high-dimensional ambient space concentrate around a lower-dimensional geometric structure.

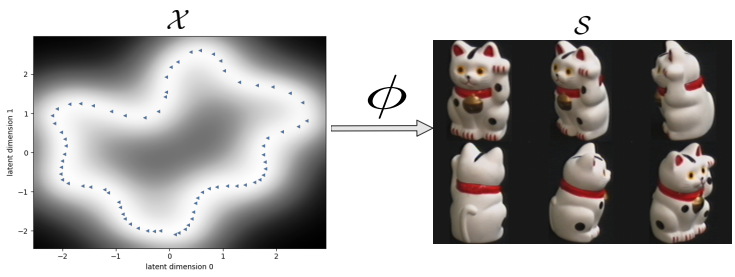


Figure 1.1: Illustration of the manifold hypothesis. Although the observations are represented in a high-dimensional ambient space, their intrinsic variability may be governed by a lower-dimensional manifold. For example, data points in $\mathbb{R}^3$ may concentrate near a two-dimensional curved surface, suggesting that the essential degrees of freedom are much lower than the ambient dimension.

In many contemporary learning pipelines, latent variable models are widely used to en-

code high-dimensional observations into lower-dimensional latent representations, which then serve as the foundation for downstream probabilistic models, including diffusion-based generative frameworks [35, 37]. Although such representations improve computational tractability and modeling efficiency, the nonlinear mappings induced by learned encoders (e.g., neural networks) can introduce substantial spatial distortion and anisotropy. As a result, the induced latent geometry is highly non-uniform and generally unknown, which can significantly affect the behavior of probabilistic models defined in the latent space, leading to biased density estimation, inefficient sampling, and degraded generative performance. This raises fundamental questions about how stochastic processes should be formulated in such spaces [3].

Since diffusion-based generative models are governed by stochastic dynamics defined over the latent space, their behavior is inherently shaped by the underlying geometry. Consequently, neglecting this induced geometry can significantly impair stochastic generative modeling: diffusion trajectories may drift away from data-supported regions, allocate probability mass to geometrically inconsistent areas, and ultimately degrade sample quality, reduce distributional coverage, and destabilise conditional generation [3, 61].

From a geometric perspective, diffusion processes are intrinsically connected to Brownian Motion and heat flow on curved spaces, which are fundamentally governed by Riemannian geometry [18]. This suggests that stochastic dynamics used for generative modeling should adapt to the intrinsic geometry of the latent representation rather than relying on Euclidean assumptions. In most practical settings, however, this geometry is unknown and must be inferred from finite and noisy data, and principled methods for incorporating such data-driven geometric information into diffusion models remain limited.

These considerations motivate the need for geometry-aware latent diffusion models that explicitly incorporate intrinsic geometric information inferred from data. This thesis addresses this gap by proposing an Intrinsic Hybrid Latent Diffusion Model (ILDM), which formulates diffusion dynamics that adapt to spatially varying geometric uncertainty in latent space. By integrating probabilistic estimates of latent geometry into the forward and reverse diffusion processes, ILDM aims to generate samples that better respect the intrinsic structure of the underlying data support and achieve more stable and consistent generative behavior.

Closely related geometric challenges also arise in regression tasks on manifold-supported data. Gaussian process (GP) models provide a principled Bayesian framework for regression and uncertainty quantification [2], but are typically constructed using Euclidean covariance structures that do not reflect intrinsic data geometry. When data are supported on unknown manifolds, such constructions may induce misleading similarity relationships and distorted uncertainty estimates [60].

This motivates the development of intrinsic Gaussian process models that respect geometry inferred from data. As a complementary contribution, this thesis introduces Gaussian Processes on Unknown Manifolds (GPUM), which extend GP regression to manifold-supported data by incorporating probabilistic geometric information into the covariance construction [30]. While generative modeling constitutes the primary focus of this thesis, GPUM addresses the parallel challenge of geometry-aware regression and further illustrates the central role of intrinsic geometry in probabilistic learning.

These considerations motivate a unified perspective in which intrinsic geometry is treated as a statistical object inferred from data and explicitly incorporated into probabilistic generative and regression models. This thesis adopts this perspective by developing geometry-aware diffusion and regression frameworks that adapt to spatially varying geometric uncertainty in latent and observation spaces.

1.2 Goal

The goal of this thesis is to develop probabilistic learning methods that explicitly account for the intrinsic geometry of data supported on unknown manifolds. Rather than relying on Euclidean assumptions or local geometric approximations, this work aims to infer manifold geometry from data and incorporate it directly into probabilistic models, thereby enabling principled stochastic dynamics, meaningful notions of similarity, and reliable uncertainty quantification.

To achieve this goal, the thesis pursues two parallel research objectives:

- **Geometry-aware generative modeling (ILDM).** The first objective is to develop geometry-consistent stochastic diffusion models for generative learning in latent spaces. Instead of assuming isotropic Euclidean noise, this line of work formulates diffusion dynamics that are aligned with the intrinsic Riemannian geometry of learned latent manifolds. By incorporating probabilistic estimates of manifold metrics into the diffusion process, the proposed ILDM aims to yield more geometrically consistent and reliable generative sampling on curved, anisotropic, or spatially distorted latent representations.

- **Intrinsic Gaussian process regression on unknown manifolds (GPUM).** The second objective is to extend Gaussian process regression to settings where data lie on nonlinear manifolds embedded in high-dimensional spaces. Rather than relying on naive substitutions of Euclidean distances with geodesic distances, this line of work develops intrinsic Gaussian process models that respect the underlying

manifold geometry inferred from data. By leveraging characterisations of manifold geometry, the proposed GPUM framework aims to provide meaningful notions of similarity and reliable uncertainty quantification for regression on manifolds.

Beyond modeling considerations, this thesis addresses the computational challenges inherent in geometry-aware probabilistic modeling on high-dimensional manifolds. In particular, it develops efficient and scalable inference and approximation techniques—including structured metric representations and stochastic approximations—that enable ILDM and GPUM to be applied to high-dimensional data and moderately large datasets while preserving essential geometric properties.

Finally, the thesis aims to validate the proposed methods through both methodological analysis and empirical evaluation. Synthetic examples are used to illustrate geometric behavior under controlled conditions, while real-world datasets are employed to demonstrate the practical advantages of incorporating intrinsic geometry into probabilistic generative modeling and regression.

1.3 Outline of the Thesis

The remainder of this thesis is organised as follows.

Chapter 2 introduces the geometric and stochastic foundations required throughout the thesis. It reviews essential concepts from Riemannian geometry, including Riemannian metrics, the Laplace–Beltrami operator, and intrinsic notions of distance. The chapter further discusses diffusion processes on manifolds, focusing on the heat equation, heat kernel, and Brownian Motion, and establishes their role in geometric inference and stochastic modeling.

Chapter 3 provides background on diffusion and score-based generative models. It reviews score matching, noise-perturbed and time-dependent score-based models, and latent diffusion frameworks. This chapter serves as a bridge between classical stochastic processes and modern generative modeling techniques.

Chapter 4 presents the proposed Intrinsic Hybrid Latent Diffusion Model. The forward and reverse diffusion processes are formulated to respect the geometry of the learned latent manifold, and classifier-free guidance is incorporated to enable controllable generation. The chapter details the model formulation, training objectives, and sampling procedures.

Chapter 5 evaluates the proposed diffusion-based models through a series of experiments on synthetic and real-world datasets, including COIL images, cardiac MRI data, and

MNIST. Quantitative metrics and qualitative visualisations are used to assess generative quality, conditional control, and the impact of incorporating intrinsic geometry.

Chapter 6 develops Gaussian process models defined on unknown manifolds. The chapter introduces geometry-aware GP priors, methods for learning manifold structure from data using latent variable models, and techniques for estimating intrinsic transition densities via Brownian Motion simulation and graph-based approximations. Experimental results on synthetic and image datasets are presented to demonstrate the effectiveness of the proposed approach.

Finally, Chapter 7 concludes the thesis and discusses potential directions for future research.

Chapter 2

Riemmanian Geometry Background and Stochastic Manifold Learning

This chapter introduces the mathematical foundations required for the geometric and probabilistic framework developed in the subsequent parts of this thesis. We begin with the elements of Riemannian geometry that describe the intrinsic structure of smooth manifolds, including tangent spaces, Riemannian metrics, and the Laplace–Beltrami operator. These notions provide a coordinate-free description of distances, angles, and curvature, forming the analytical basis for studying processes on curved spaces.

The heat equation and its associated heat kernel describe the diffusion of information over a manifold and encode fundamental geometric properties of the underlying space. Their probabilistic counterpart is Brownian motion on a manifold, whose transition densities coincide with the heat kernel, providing a natural stochastic interpretation of geometric diffusion processes.

The chapter concludes by establishing the probabilistic interpretation of diffusion processes on manifolds, which provides a fundamental link between Riemannian geometry and stochastic dynamics on curved spaces. These theoretical foundations motivate the probabilistic mapping framework introduced in the next chapter, where the geometry of an unknown manifold is inferred directly from high-dimensional observations.

2.1 Foundations of Riemannian Geometry

This section develops the geometric foundations of Riemannian geometry. The discussion is grounded in Riemannian geometry [1], which generalises Euclidean geometry to smooth curved spaces. By equipping a differentiable manifold with a smoothly varying inner

product on its tangent spaces, Riemannian geometry provides intrinsic notions of distance, angles, geodesics, curvature, and volume [6, 7]. These notions enable a rigorous analysis of geometric phenomena independent of any particular embedding of the manifold.

Formally, a differentiable manifold $\mathcal{M}$ of dimension d is a topological space that locally resembles $\mathbb{R}^d$. Each point $x \in \mathcal{M}$ possesses an associated tangent space $T_x\mathcal{M}$, a d-dimensional vector space representing the first-order linear approximation of the manifold around x. One way to characterise the tangent space $T_x\mathcal{M}$ is via smooth curves: for any smooth curve $\gamma : (-\epsilon, \epsilon) \to \mathcal{M}$ satisfying $\gamma(0) = x$, the derivative $\dot{\gamma}(0)$ defines a velocity vector at x, and the set of all such velocity vectors forms the tangent space $T_x\mathcal{M}$.

Geometrically, the tangent space captures all admissible instantaneous directions of smooth curves passing through x. Although this abstraction is natural from a differential-geometric viewpoint, it also aligns with common intuitions in data analysis: the tangent space characterises the locally linear structure implicitly present in high-dimensional observations.

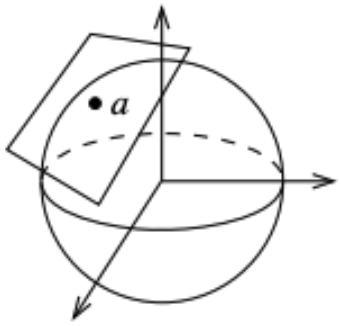


Figure 2.1: Illustration of a tangent space $T_a\mathcal{M}$ at a point a on the sphere. Adapted from Lee [8].

This geometric intuition is illustrated in Fig. 2.1, where a tangent plane is shown attached to a point on the sphere. A smooth curve passing through the point has a velocity vector—its derivative—lying entirely within the tangent plane.

Definition 2.1 (Riemannian metric)**.** *A* Riemannian metric *on a differentiable manifold $\mathcal{M}$ is a smoothly varying inner product*

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

assigning to each tangent space a positive-definite bilinear form. The collection $\{g_x\}_{x\in\mathcal{M}}$ induces a rich geometric structure, including notions of length and angle for tangent vectors, lengths of curves, and distances between points. When $\mathcal{M} = \mathbb{R}^d$ with the standard Euclidean inner product, the resulting geometry reduces to the classical Euclidean case.

In many applications, data-generating processes often result in high-dimensional data samples, which, when considered collectively, effectively form low-dimensional manifolds embedded within high-dimensional Euclidean spaces. For instance, images of faces under varying lighting conditions may, to a good approximation, lie on a lower-dimensional manifold embedded within the high-dimensional image space. A common approach to modeling such a manifold $\mathcal{M} \subset \mathbb{R}^p$, where p denotes the dimensionality of the ambient (observation) space, is through a probabilistic mapping function $\phi : \mathbb{R}^q \to \mathcal{M}$, where $\mathbb{R}^q$ denotes a q-dimensional latent space representing the intrinsic coordinates of the manifold, with $q \ll p$. This mapping aims to parametrise the manifold, allowing points on $\mathcal{M}$ to be represented by lower-dimensional coordinates in $\mathbb{R}^q$. However, in most practical scenarios, neither the explicit form of the manifold $\mathcal{M}$ nor the mapping φ is known, making it necessary to infer the intrinsic geometric structure of $\mathcal{M}$ directly from data samples.

A classical illustration is the Swiss roll in $\mathbb{R}^3$. A two-dimensional patch of the surface can be described by a chart, that is, a smooth coordinate map $\varphi \colon U \subset \mathbb{R}^2 \to \mathcal{M}$, where the two latent variables provide an unrolled representation of the geometry.

For such a smooth parameterisation φ, the induced Riemannian metric g on $\mathcal{M}$ can be expressed in local coordinates on U via its matrix representation

$$g(x) = \mathbf{J}_\varphi(x)^\top \mathbf{J}_\varphi(x),$$

where $g(x)$ is the matrix representation of the metric tensor g_x in the coordinate chart φ, and $\mathbf{J}_\varphi(x)$ denotes the Jacobian matrix of φ at x, with entries

$$[\mathbf{J}_\varphi(x)]_{i,j} = \frac{\partial \varphi_i}{\partial x_j}(x).$$

Equivalently, the metric components introduced above satisfy

$$g_{i,j}(x) = \left\langle \frac{\partial \varphi}{\partial x_i}, \frac{\partial \varphi}{\partial x_j} \right\rangle.$$

This induced form explicitly shows how local inner products in the latent space are pulled back from the ambient Euclidean geometry. Integrating these local measurements yields global geometric quantities on the manifold. This parametrisation also enables the computation of local geometric properties (e.g., distances, angles) directly in the latent coordinates, as illustrated in Fig. 2.2, where blue triangles in the chart are mapped to black points on the Swiss roll.

The Riemannian metric g must satisfy three key properties to consistently encode geometric structure and ensure a well-defined inner product on tangent spaces:

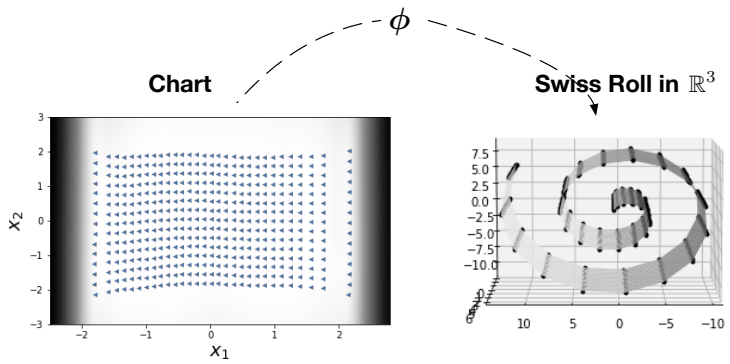


Figure 2.2: An illustration of a parameterisation of the manifold $\mathcal{M}$ (Swiss roll), estimated from a data cloud from right, adapted from [30]. The map ϕ sends the chart into $\mathcal{M} \subset \mathbb{R}^3$. The blue triangles in the chart are mapped to the black dots on $\mathcal{M} \subset \mathbb{R}^3$.

Proposition 2.1 (Properties of a Riemannian metric)**.** *Let $(\mathcal{M}, g)$ be a Riemannian manifold. Then for each $x \in \mathcal{M}$, the inner product g_x on $T_x\mathcal{M}$ satisfies:*

1. ***Symmetry:*** *$g_x(u, v) = g_x(v, u)$ for all $u, v \in T_x\mathcal{M}$.*
2. ***Positive-definiteness:*** *$g_x(v, v) > 0$ whenever $v \in T_x\mathcal{M}$ is nonzero.*
3. ***Smoothness:*** *In any coordinate chart, the component functions $g_{ij}(x)$ of g are smooth functions of x.*

These local properties give rise to global geometric objects such as geodesics, curvature tensors, and volume forms. Of particular importance to this thesis is the differential operator intrinsically defined by the metric: the Laplace–Beltrami operator. It generalises the classical Laplacian to curved spaces and serves as a fundamental analytical tool in geometry and partial differential equations. Its formal definition in local coordinates is given in Definition 2.2.

Definition 2.2 (Laplace–Beltrami operator)**.** *Let $(\mathcal{M}, g)$ be a Riemannian manifold with metric tensor g. The* Laplace–Beltrami operator *Δ_s is the intrinsic analogue of the classical Laplacian.*

In local coordinates $(x^1, \ldots, x^q)$, where superscripts denote coordinate indices, it is expressed as

$$\Delta_s f = \frac{1}{\sqrt{G}} \frac{\partial}{\partial x^j} \left(\sqrt{G}\, g^{ij} \frac{\partial f}{\partial x^i} \right),$$

where $G = \det(g(x))$ denotes the determinant of the metric matrix in local coordinates, g^{ij} denotes the (i, j)-th entry of the inverse metric tensor, and f is a smooth function on $\mathcal{M}$.

The Laplace–Beltrami operator retains the key features of the standard Laplacian in Euclidean space: it acts linearly on functions, is elliptic, and is self-adjoint with respect to the natural L^2 inner product induced by the Riemannian volume form. In the special case where $\mathcal{M} = \mathbb{R}^q$ with the standard Euclidean metric, it reduces to the ordinary Laplacian.

Crucially, for data-defined manifolds where the explicit metric g and parameterisation are often unknown, the Laplace–Beltrami operator can be effectively approximated directly from data samples, making it a foundational tool for geometric inference. A central role of the Laplace–Beltrami operator lies in defining diffusion processes on manifolds. Analogous to the Euclidean setting where the Laplacian governs the classical heat equation, its manifold counterpart generates the intrinsic heat flow, whose fundamental solution is the heat kernel [9, 10]. The heat kernel, as the solution to the heat equation describing heat diffusion over time on the manifold, encodes rich information about the manifold's geometry and connectivity, offering a powerful tool to analyse the intrinsic structure of manifolds—even when only observed through data samples.

2.1.1 Heat Equation and Heat Kernel

Having introduced the Laplace–Beltrami operator, we now examine its role as the generator of diffusion processes on Riemannian manifolds. The heat equation provides a natural analytic framework for describing how a quantity such as temperature evolves over time on a curved space, with the manifold's geometry encoded directly through the metric and the associated Laplace–Beltrami operator.

Consider the heat equation on a Riemannian manifold $\mathcal{M}$, formally expressed as:

$$\frac{\partial}{\partial t} K_{\text{heat}}(s_0, s, t) = \frac{1}{2} \Delta_s K_{\text{heat}}(s_0, s, t), \quad s_0, s \in \mathcal{M},$$

where Δ_s denotes the Laplace–Beltrami operator acting on the spatial variable s, and $K_{\text{heat}}(s_0, s, t)$ denotes the heat kernel, representing the distribution of heat at point s and time t given an initial heat source located at s_0.

A heat kernel of $\mathcal{M}$ is therefore a smooth function $K(s_0, s, t)$ on $\mathcal{M} \times \mathcal{M} \times \mathbb{R}^+$ that solves the heat equation and captures how heat propagates across the manifold. It must satisfy the initial condition

$$\lim_{t \to 0} K_{\text{heat}}(s_0, s, t) = \delta(s_0, s),$$

where $\delta(s_0, s)$ is the Dirac delta function centered at s_0. This condition states that as time approaches zero, the heat kernel concentrates all heat at the initial point, reflecting the

initial heat source.

When the manifold has a boundary $\partial\mathcal{M}$, the heat kernel becomes uniquely determined only after specifying a boundary condition. A commonly imposed condition is the Neumann boundary condition

$$\frac{\partial K}{\partial n} = 0 \quad \text{along } \partial\mathcal{M},$$

where n denotes the outward-pointing normal vector. This condition ensures that there is no heat flux across the boundary, meaning that heat is contained within the manifold. The Neumann boundary condition implies that the boundary is insulated [11].

If $\mathcal{M}$ is the Euclidean space $\mathbb{R}^q$, the heat kernel has a well-known closed-form solution, which takes the form of a time-varying Gaussian function:

$$K_{\text{heat}}(s_0, s, t) = \frac{1}{(2\pi t)^{q/2}} \exp\left(-\frac{||s_0 - s||^2}{2t}\right), \quad s \in \mathbb{R}^q.$$

Here, $||s_0 - s||$ represents the Euclidean distance between points s_0 and s. The diffusion time t controls the rate of decay and governs the variance of the Gaussian distribution. As t increases, the Gaussian spreads out, representing the diffusion of heat away from the initial point. This provides an intuitive understanding of how the heat kernel behaves in Euclidean space.

We write $K^t_{\text{heat}}(s_0, s)$ for $K_{\text{heat}}(s_0, s, t)$. The heat kernel is a powerful tool for characterising manifold geometry. Its short-time asymptotic behavior (as $t \to 0$) encodes detailed information about curvature, volume, and local invariants, while its long-time behavior (as $t \to \infty$) reflects global connectivity and the spectral structure of the Laplace–Beltrami operator. This dual local-global nature underlies a broad range of applications in geometric analysis and motivates its use in data-driven manifold learning and diffusion-based methods.

The heat equation provides a powerful theoretical framework for understanding diffusion processes on Riemannian manifolds. Its solution, the heat kernel, encodes intrinsic geometric properties of the manifold, such as curvature and connectivity. However, analytical solutions, particularly on manifolds with complex geometries or boundary conditions, are often intractable. In these cases, numerical and approximation methods are commonly employed to estimate the heat kernel and analyse the behaviour of diffusion. An alternative and deeply insightful perspective on heat diffusion arises from its intimate relationship with stochastic processes. Lawler [15] provides a comprehensive discussion of the fundamental probabilistic connections between Brownian motion and the heat equation, illustrating this strong relationship. The subsequent subsection will delve into the definition

and properties of Brownian motion on Riemannian manifolds, outlining its fundamental role in both theoretical analysis and practical applications for geometric inference.

2.1.2 Brownian Motion

Brownian motion (BM), also known as a Wiener process, provides a fundamental probabilistic model for describing random motion. While its classical origin lies in the physical observation of particles suspended in fluid, its mathematical formulation has become central to modern probability theory, partial differential equations, and geometric analysis. On a Riemannian manifold, BM offers a stochastic counterpart to the analytic framework established by the heat equation, providing an alternative yet equivalent perspective on diffusion [15, 18]. In this setting, Brownian motion B_t is defined as a continuous-time stochastic process taking values in $\mathcal{M}$, whose behavior reflects the underlying geometric structure through the Riemannian metric.

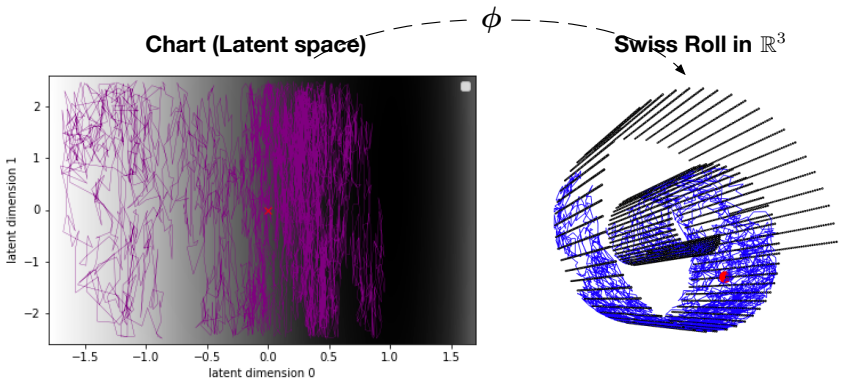


Figure 2.3: A BM sample path (blue line, right panel) on $\mathcal{M}$ (Swiss roll in $\mathbb{R}^3$) and its equivalent stochastic process (purple line, left panel) in the chart (latent space) in $\mathbb{R}^2$, adapted from our prior work [30]. The map $\phi : \mathbb{R}^2 \to \mathcal{M} \subset \mathbb{R}^3$ is a parametrisation of $\mathcal{M}$. The purple path in the chart is simulated using (2.8), and its image under ϕ is the blue path on the Swiss roll. The red dot marks the starting location of the BM trajectory. The horizontal axis of the latent space represents the radius of the Swiss roll, while the vertical axis corresponds to its angle. The grey shading indicates the magnification factor: darker regions in the latent space are stretched more when mapped to $\mathcal{M} \subset \mathbb{R}^3$.

Figure 2.3 shows three simulated Brownian motion trajectories on a two-dimensional domain.

We recall several standard properties of BM on a Riemannian manifold [17]:

Proposition 2.2. *Let $\mathcal{M}$ be a Riemannian manifold, and let B_t denote a BM on $\mathcal{M}$. The following properties hold:*

1. ***Initial Condition:*** *$B_0 = s_0$, where $s_0 \in \mathcal{M}$. The process starts at a prescribed*

point on the manifold.

2. ***Independent Increments:*** *For any $0 \leq t_1 < t_2 < t_3 < t_4$, the increments $B_{t_2} - B_{t_1}$ and $B_{t_4} - B_{t_3}$ are independent. The future evolution of the process is independent of its past.*

3. ***Gaussian Increments:*** *For $t > 0$ and infinitesimal $dh > 0$, the increment $B_{t+dh} - B_t$, when expressed in the tangent space $T_{B_t}\mathcal{M}$ using a normal coordinate chart, follows a Gaussian distribution whose covariance is determined by the Riemannian metric g at the point B_t.*

4. ***Continuous Paths:*** *The sample paths of BM are almost surely continuous functions of time. In other words, the process moves smoothly across the manifold and does not undergo instantaneous jumps.*

The deep connection between BM and the heat equation lies in the fact that the transition density of BM, representing the probability of finding a Brownian path at position s at time t given it originated at s_0, is precisely equivalent to the heat kernel $K_{\text{heat}}(s_0, s, t)$ [10, 17]. This fundamental equivalence, derived from the relationship between the Laplace–Beltrami operator and the infinitesimal generator of BM, provides a powerful stochastic interpretation of heat diffusion. While introduced here to establish this crucial link between probabilistic and analytic approaches, a more detailed exploration of the transition density's properties, computation, and specific applications will be provided in Subsection 6.6.

Furthermore, the Laplace–Beltrami operator, a key component of the heat equation, also serves as the infinitesimal generator of BM on a Riemannian manifold [17]. Let $(\mathcal{M}, \langle\cdot,\cdot\rangle_x)$ be a Riemannian manifold. For a smooth function $f : \mathcal{M} \to \mathbb{R}$, its gradient $\operatorname{grad} f$ is the unique vector field satisfying

$$\langle \operatorname{grad} f, X\rangle_x = X(f), \qquad X \in T_x\mathcal{M}.$$

where $X(f)$ denotes the directional derivative of f along the vector field X.

Throughout all coordinate expressions, we adopt the Einstein summation convention: whenever an index appears once as an upper index and once as a lower index in a monomial, summation over its entire range is implied, e.g.,

$$A^i B_i \;:=\; \sum_{i=1}^{n} A^i B_i. \tag{2.1}$$

The Laplace–Beltrami operator is defined as the divergence of the gradient:

$$\Delta_{\mathcal{M}} f = \operatorname{div}(\operatorname{grad} f).$$

In local coordinates, using Einstein summation, this becomes

$$\Delta_{\mathcal{M}} f = \frac{1}{\sqrt{G}} \frac{\partial}{\partial x^i} \left(\sqrt{G}\, \mathcal{G}^{ij} \frac{\partial f}{\partial x^j} \right), \tag{2.2}$$

where $\mathcal{G}^{ij}$ are the components of the inverse metric tensor and $G = \det(\mathcal{G}_{ij})$.

Using the product rule, this expression expands to

$$\begin{aligned} \Delta_{\mathcal{M}} f &= \frac{1}{\sqrt{G}} \frac{\partial}{\partial x^i} \left(\sqrt{G}\, \mathcal{G}^{ij} f_j \right) && (2.3) \\ &= \mathcal{G}^{ij} f_{ij} + \frac{1}{\sqrt{G}} \left(\frac{\partial \sqrt{G}}{\partial x^i}\, \mathcal{G}^{ij} + \sqrt{G} \frac{\partial \mathcal{G}^{ij}}{\partial x^i} \right) f_j && (2.4) \\ &= \mathcal{G}^{ij} f_{ij} + b^j f_j, && (2.5) \end{aligned}$$

where the drift vector field is

$$b^j = \frac{1}{\sqrt{G}} \frac{\partial}{\partial x^i} \left(\sqrt{G}\, \mathcal{G}^{ij} \right). \tag{2.6}$$

To highlight the drift component of the induced Brownian motion, we collect the coefficients of the first-order derivatives and define the drift vector field:

$$\Delta_{\mathcal{M}} f = \mathcal{G}^{ij} f_{ij} + b^j f_j. \tag{2.7}$$

This decomposition is fundamental in stochastic analysis on manifolds, because $\Delta_{\mathcal{M}}$ is the infinitesimal generator of Brownian motion on $\mathcal{M}$. As shown in Hsu [17, 18], the Brownian motion in local coordinates admits the Itô representation

$$dx^i(t) = \frac{1}{2} \sum_{j=1}^{q} \frac{\partial}{\partial x^j} \left(\mathcal{G}_{ij}^{-1} \sqrt{G} \right) \frac{1}{\sqrt{G}} dt + \left(\mathcal{G}^{-1/2} dB(t) \right)_i, \tag{2.8}$$

where $\mathcal{G}$ is the metric tensor of $\mathcal{M}$, $B(t)$ is Euclidean Brownian motion and $g^{-1/2}$ denotes a matrix square root of the inverse metric tensor. This stochastic differential equation (SDE) provides the basis for numerical simulations of Brownian motion and heat kernel approximation on a general Riemannian manifold.

Here, $x(t)$ represents the local coordinates on the manifold, while $g = (g_x)$ represents

the Riemannian metric tensor, encapsulating the intrinsic geometry of the manifold. The term $B(t)$ represents an independent Brownian motion in Euclidean space, serving as the stochastic driving force. Additionally, g^{ij} denotes the components of the inverse metric tensor, and $(g^{-1/2}dB(t))^i$ represents the i-th component of the stochastic differential. The discrete form of this SDE is crucial for numerical simulations.

The discrete form of this SDE is crucial for numerical simulations. The discretised version of Equation (2.8) is presented as follows:

$$\begin{aligned} x^i(t) = x^i(t-\Delta t) &+ \frac{1}{2}\sum_{j=1}^{q}\left(-\mathcal{G}^{-1}\frac{\partial \mathcal{G}}{\partial x^j}\mathcal{G}^{-1}\right)_{ij}\Delta t \\ &+ \frac{1}{4}\sum_{j=1}^{q}\left(\mathcal{G}^{-1}\right)_{ij}\operatorname{tr}\left(\mathcal{G}^{-1}\frac{\partial \mathcal{G}}{\partial x^j}\right)\Delta t + \left(\sqrt{\Delta t}\mathcal{G}^{-1/2}z_q\right)_i \end{aligned} \tag{2.9}$$

or, for practical computational efficiency, equivalently:

$$x^i(t) = \mu(x^i(t-\Delta t), \Delta t)_i + \left(\sqrt{\Delta t}\mathcal{G}^{-1/2}z_q\right)_i, \tag{2.10}$$

where $\mu(x, \Delta t)$ collects the deterministic drift terms in 2.9, i.e., the terms involving g^{-1} and its derivatives.

Here, Δt represents the diffusion time, or the time step, for the BM simulation, and z_q is a q-dimensional standard normal random variable. The discrete form of the SDE also naturally defines a proposal mechanism for simulating BM on the manifold. In particular, Eq. (2.10) specifies a Gaussian transition kernel

$$q\big(x(t) \mid x(t-\Delta t)\big) = \mathcal{N}\big(x(t) \,\big|\, \mu(x(t-\Delta t), \Delta t),\, \Delta t\, \mathcal{G}^{-1}(x(t-\Delta t))\big), \tag{2.11}$$

which serves as the proposal distribution for advancing the Brownian motion from time $t - \Delta t$ to t. This formulation explicitly incorporates the local Riemannian geometry through the metric tensor $\mathcal{G}(x)$, ensuring that the simulated trajectories respect the intrinsic structure of the underlying manifold.

Equations (2.8), (2.9), and (2.10) collectively illustrate how the geometry of $\mathcal{M}$ governs local diffusion. The metric tensor influences both drift and diffusion term, and the resulting stochastic dynamics provide insight into the manifold's intrinsic structure. Consequently, the analysis of BM and its trajectories offers a powerful tool for estimating geometric quantities, studying the heat kernel, and understanding spectral properties associated with the Laplace–Beltrami operator. The stochastic viewpoint thus complements the analytic framework introduced earlier, forming a robust foundation for the geometric methods developed in subsequent chapters.

2.2 Probabilistic Mapping Function on Latent Manifolds

A central difficulty in manifold-based learning is that the geometric structure of the underlying space is not directly observable but must instead be reconstructed from high-dimensional data. The preceding sections have introduced the analytical machinery required to study stochastic processes and differential operators on a Riemannian manifold—most notably the Laplace–Beltrami operator, the heat equation and Brownian motion. These developments implicitly assume that the manifold and its Riemannian metric are available in analytic form. In practical data-driven settings, however, the manifold is not given a priori; only a finite collection of points embedded in Euclidean space is observed, and the metric becomes an object that must itself be inferred.

To bridge this gap, the present section introduces a probabilistic framework for recovering local geometric structure directly from data using Gaussian Processes (GPs). The key idea is to view the manifold as the image of an unknown smooth map from a low-dimensional latent domain into the ambient space. By placing a GP prior on this mapping, we obtain a flexible nonlinear generative model capable of capturing complex embeddings while simultaneously quantifying uncertainty in both the function values and their derivatives. This uncertainty propagates naturally to the induced Riemannian metric tensor, providing a principled means of assessing the reliability of the inferred geometry. The resulting GP-based characterisation of the metric forms the basis for constructing heat kernels and developing geometry-aware stochastic processes in the subsequent chapters.

2.2.1 Latent Variable Model and GP-Based Mapping

Building on the probabilistic perspective outlined above, we now introduce the latent-variable formulation underlying the GP-based mapping. Let $\mathcal{S} = \{s_i \in \mathbb{R}^p \mid i = 1, \dots, n+u\}$ denote the observed dataset. Although the ambient dimension p is high, we assume that the samples lie on, or at least near, a smooth q-dimensional manifold $\mathcal{M} \subset \mathbb{R}^p$, where $q \ll p$. To describe the intrinsic structure, we introduce latent variables

$$\mathcal{X} = \{x_i \in \mathbb{R}^q \mid i = 1, \dots, n+u\},$$

which are interpreted as intrinsic coordinates on the manifold. The manifold itself is assumed to be generated by a smooth function

$$\phi : \mathbb{R}^q \to \mathbb{R}^p,$$

which maps latent coordinates into the observation space.

Figure 2.4 visualises the relationship between the latent space $\mathcal{X}$ and the observation space $\mathcal{S}$ for a subset of the COIL image dataset. Each point in the latent space corresponds to an intrinsic coordinate x_i, while the mapping $\phi : \mathcal{X} \to \mathcal{S}$ transforms these latent coordinates into high-dimensional observations. The left sub-figure shows the latent representation, where the background shading encodes the predictive variance of the GP mapping, regions farther away from observed data exhibit higher uncertainty. The right sub-figure shows several example images s_i that result from evaluating ϕ at the corresponding latent positions. This visualisation highlights the central idea of our model: the manifold geometry in the latent space is implicitly defined through the smooth mapping ϕ that generates the observed data.

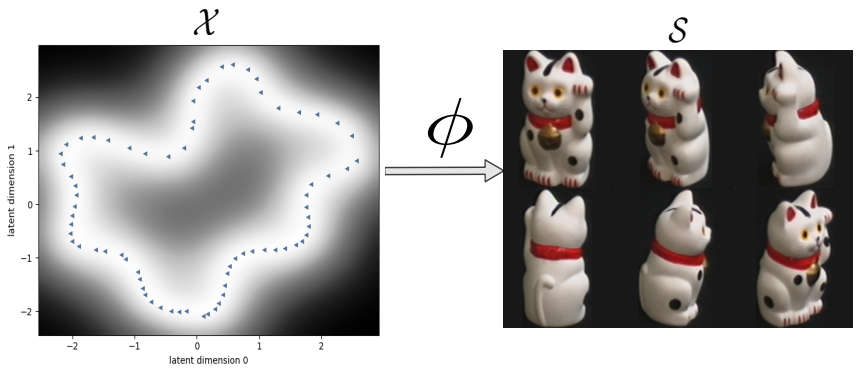


Figure 2.4: Illustration of manifold parameterisation for the COIL image dataset. The mapping $\phi : \mathcal{X} \to \mathcal{S}$ projects latent points (blue triangles in the left sub-figure) to their corresponding images (six examples shown in the right sub-figure). Background shading in the latent space represents the uncertainty of ϕ, quantified as the variance of the mapping, which increases with distance from observed data points.

Each observation s_i is modelled as a noisy evaluation of ϕ,

$$s_i^j = \phi^j(x_i) + e_i^j, \qquad e_i^j \sim \mathcal{N}(0, \beta^2), \tag{2.12}$$

for $j = 1, \ldots, p$. The independent Gaussian noise captures measurement noise and small deviations from the ideal manifold model.

A crucial assumption of this work is that each coordinate function ϕ^j is endowed with an independent Gaussian Process prior. This independence assumption greatly simplifies the analysis while remaining sufficiently expressive to capture highly non-linear embeddings. Concretely, we assume:

$$\phi^j \sim \mathcal{GP}(0, k(\cdot, \cdot)), \qquad j = 1, \ldots, p,$$

where k denotes a positive-definite covariance kernel. The GP prior provides several advantages: it ensures smoothness of the mapping, yields closed-form posterior distributions, and permits analytical computation of derivatives and their uncertainties. These properties will be essential when we derive the local Riemannian metric associated with the learned manifold.

2.2.2 Riemannian Geometry Induced by the GPLVM

The Gaussian Process Latent Variable Model (GPLVM), originally proposed by Lawrence [16, 48], defines a probabilistic nonlinear mapping

$$\phi : \mathbb{R}^q \to \mathbb{R}^p,$$

from a low-dimensional latent space to the observation space by placing independent Gaussian Process (GP) priors over each output dimension. Beyond its role as a nonlinear dimensionality reduction method, the GPLVM implicitly endows the latent space with a Riemannian geometric structure through this mapping. In this section, we derive the Jacobian and the induced Riemannian metric associated with the GPLVM, which form the foundation for the geometric constructions developed throughout this thesis.

GPLVM Mapping and Jacobian. In the GPLVM, observations are generated according to $s_i^j = \phi^j(x_i) + e_i^j$, where each coordinate function ϕ^j is assigned an independent GP prior,

$$\phi^j \sim \mathcal{GP}(0, k(\cdot,\cdot)), \qquad j = 1, \dots, p.$$

The local linearisation of the mapping at a latent point $x \in \mathbb{R}^q$ is described by the Jacobian matrix

$$\mathbf{J}(x) = \frac{\partial \phi(x)}{\partial x} \in \mathbb{R}^{p \times q}.$$

Its transpose can be written explicitly as

$$\mathbf{J}(x)^\top = \begin{bmatrix} \frac{\partial \phi^1(x)}{\partial x^1} & \cdots & \frac{\partial \phi^p(x)}{\partial x^1} \\ \vdots & \ddots & \vdots \\ \frac{\partial \phi^1(x)}{\partial x^q} & \cdots & \frac{\partial \phi^p(x)}{\partial x^q} \end{bmatrix}. \tag{2.13}$$

The Jacobian defines the differential map between the latent and observation spaces and determines how the Euclidean geometry in $\mathbb{R}^p$ is pulled back onto the latent space.

Posterior Distribution of the Jacobian. A key advantage of the GPLVM is that Gaussian processes admit closed-form joint distributions over function values and derivatives [2]. As a result, the posterior distribution of the Jacobian can be derived analytically.

For a single output dimension j, the joint prior over function values and derivatives at a test point x_* is given by

$$\begin{bmatrix} \phi(\mathcal{X})^j \\ \dfrac{\partial \phi(x_*)^j}{\partial x_*} \end{bmatrix} \sim \mathcal{N}\left(0,\ \begin{bmatrix} K_{\mathcal{X}\mathcal{X}} & \partial K_{\mathcal{X},*} \\ \partial K_{\mathcal{X},*}^\top & \partial^2 K_{*,*} \end{bmatrix}\right), \tag{2.14}$$

where $K_{\mathcal{X}\mathcal{X}}$ is the kernel matrix evaluated at the latent points $\mathcal{X}$, $\partial K_{\mathcal{X},*}$ denotes the first derivative of the kernel with respect to x_*, and $\partial^2 K_{*,*}$ denotes the corresponding second derivative.

Conditioning on the observed data yields the posterior distribution of the Jacobian rows,

$$p(\mathbf{J}(x_*) \mid \mathcal{X}, \mathcal{S}) = \prod_{j=1}^{p} \mathcal{N}\big(\mu_{\mathbf{J}}^j(x_*),\ \Sigma_{\mathbf{J}}(x_*)\big)\,, \tag{2.15}$$

with

$$\mu_{\mathbf{J}}^j = \partial K_{\mathcal{X},*}^\top K_{\mathcal{X}\mathcal{X}}^{-1} \mathbf{s}_:^j, \qquad \Sigma_{\mathbf{J}} = \partial^2 K_{*,*} - \partial K_{\mathcal{X},*}^\top K_{\mathcal{X}\mathcal{X}}^{-1} \partial K_{\mathcal{X},*},$$

where $s_:^j \in \mathbb{R}^n$ denotes the vector of observations corresponding to the j-th component evaluated at all training inputs. Thus, each row of the Jacobian follows a multivariate Gaussian distribution, and uncertainty in the GP mapping propagates directly to uncertainty in the local geometry.

Induced Riemannian Metric. The GPLVM induces a Riemannian metric on the latent space via the pullback of the Euclidean metric in $\mathbb{R}^p$:

$$\mathbf{g}(x) = \mathbf{J}(x)^\top \mathbf{J}(x).$$

Since $\mathbf{J}(x)$ is a random matrix with Gaussian-distributed rows, the metric tensor is itself a random positive semidefinite matrix. In particular,

$$\mathbf{g}(x) \sim \mathcal{W}_q\big(p,\ \Sigma_{\mathbf{J}}(x),\ \mathbb{E}[\mathbf{J}(x)]^\top \mathbb{E}[\mathbf{J}(x)]\big)\,, \tag{2.16}$$

where $\mathcal{W}_q(p, \Sigma, \Theta)$ denotes the q-dimensional non-central Wishart distribution with p degrees of freedom, covariance Σ, and non-centrality parameter Θ.

Taking expectations yields a deterministic estimate of the local metric:

$$\mathcal{G}(x) = \mathbb{E}[\mathbf{g}(x)] = \mathbb{E}[\mathbf{J}(x)]^\top \mathbb{E}[\mathbf{J}(x)] + p\,\Sigma_{\mathbf{J}}(x). \tag{2.17}$$

The first term captures the geometry implied by the mean GPLVM mapping, while the second term accounts for epistemic uncertainty arising from limited or noisy data.

Metric Derivatives and Geometric Operators. The expected metric $\mathcal{G}(x)$ uniquely determines the Levi–Civita connection on the latent manifold [6, 84]. Its derivatives with respect to latent coordinates are given by

$$\frac{\partial \mathcal{G}}{\partial x^l} = \frac{\partial \mathbb{E}[\mathbf{J}]^\top}{\partial x^l}\mathbb{E}[\mathbf{J}] + \mathbb{E}[\mathbf{J}]^\top \frac{\partial \mathbb{E}[\mathbf{J}]}{\partial x^l} + p\,\frac{\partial \Sigma_{\mathbf{J}}}{\partial x^l}. \tag{2.18}$$

These quantities are required for computing Christoffel symbols, geodesics, and diffusion processes such as Brownian motion on the learned manifold.

In summary, the GPLVM provides a principled probabilistic framework in which latent geometry emerges naturally from the GP mapping. The availability of closed-form expressions for Jacobians, metric tensors, and their derivatives makes the GPLVM particularly well suited for geometric inference and stochastic processes on data-driven manifolds.

Algorithm 1 summarises the procedure for simulating Brownian motion trajectories on the implicitly learned manifold, where the local Riemannian metric is inferred from data and an uncertainty-based boundary is enforced to restrict diffusion to geometrically reliable regions.

Algorithm 1 Simulating BM sample paths on $\mathcal{M}$

Learn the metric $\mathcal{G}$ from the point cloud $\mathcal{S} = \{s_i \mid i = 1, \ldots, n+v\}$. {use eqn (2.17)}
1.1 Generate BM trajectories on implicit manifold.
for $i = 1, \ldots, n$ {n is the size of data points} **do**
  for $j = 1, \ldots, N_{BM}$ {N_{BM} is No. of trajectories} **do**
    for $l = 1, \ldots, N_t$ {N_t steps, $N_t \Delta t \to$ max diffusion time} **do**
      do {keep proposing x until it falls inside the boundary}
      $q(x_{i,j}(l) \mid x_{i,j}(l-1)) = \mathcal{N}\big(x_{i,j}(l) \mid \mu(x_{i,j}(l-1), \Delta t),\ \Delta t\, \mathcal{G}^{-1}\big)$ {use eqn (2.11)}
      while $\mathrm{Var}(\phi(x_{i,j})) > \alpha$ **or** $x_{i,j}$ is outside the boundary $\partial\mathcal{M}$ {use eqn (2.25)}
    end for
  end for
end for
return x

2.2.3 Predictive Geometry under GPLVM-Based Models

The geometric quantities derived in the previous section characterise the local structure of the latent space induced by the GPLVM mapping at locations supported by the observed data. In practice, however, many geometric operations—such as diffusion, geodesic computation, or intrinsic Gaussian process regression—require evaluating the latent geometry at arbitrary points in the latent space. This necessitates a predictive formulation of geometry that accounts for uncertainty in the learned generative mapping.

Predictive Formulation. Given the learned GPLVM and conditioning on the latent training inputs $\mathcal{X}$ and observations $\mathcal{S}$, the GP mapping admits a standard predictive distribution at any test latent location x_*. For each output dimension j, the predictor is given by

$$p\big(\phi^j(x_*) \mid \mathcal{X}, \mathcal{S}\big) = \mathcal{N}\big(k_{\mathcal{X},*}^\top K_{\mathcal{X}\mathcal{X}}^{-1}\mathbf{s}^j,\ k_{*,*} - k_{\mathcal{X},*}^\top K_{\mathcal{X}\mathcal{X}}^{-1} k_{\mathcal{X},*}\big)\,, \tag{2.19}$$

where $k_{\mathcal{X},*}$ denotes the kernel evaluations between training latent points and x_*. We consider the predictive distribution at a latent query location $x_* \in \mathbb{R}^q$.

Since differentiation is a linear operator, the same predictive mechanism extends directly to the Jacobian of the mapping. In particular, the predictor for the Jacobian is obtained by replacing kernel evaluations with their derivatives, yielding a Gaussian predictive distribution for $\mathbf{J}(x_*)$.

Predictive Geometry under the GPLVM. Under this predictive formulation, the Jacobian $\mathbf{J}(x_*)$ follows a Gaussian distribution whose mean and covariance depend smoothly on x_* through kernel derivatives. As a direct consequence of the GP predictive distribution, both the mapping and its derivatives admit well-defined posterior predictors at any latent test location x_*. In particular, the Jacobian $\mathbf{J}(x_*)$ follows a Gaussian distribution whose mean and covariance depend smoothly on x_* via kernel derivatives.

The predictive Jacobian induces a distribution over the Riemannian metric tensor

$$\mathbf{g}(x_*) = \mathbf{J}(x_*)^\top \mathbf{J}(x_*),$$

thereby defining a probabilistic metric on the latent space. This formulation captures increasing geometric uncertainty in regions that are weakly supported by data, a property that plays a central role in uncertainty-aware geometric models. In practice, the expected metric $\mathcal{G}(x_*)$ provides a stable and analytically tractable representation of the local geometry, while still reflecting epistemic uncertainty through its dependence on the predictive Jacobian covariance.

Extension to Bayesian GPLVMs. In Bayesian GPLVMs, uncertainty is further introduced through a posterior distribution over latent variables. Rather than conditioning geometric quantities on fixed latent coordinates, the predictive geometry is obtained by marginalizing over latent uncertainty. Conceptually, this leads to a geometry that reflects both uncertainty in the generative mapping and uncertainty in latent representations.

While closed-form expressions are generally no longer available, the resulting predictive metric can be approximated using variational expectations or Monte Carlo estimates. This probabilistic view of geometry, in which the metric itself is treated as a random object, aligns with recent developments in intrinsic Gaussian process models on learned manifolds, where uncertainty in the underlying geometry is an essential modeling component.

2.2.4 Bayesian-GPLVM

While the GPLVM provides a flexible non-linear mapping from latent coordinates to the observation space, its deterministic treatment of the latent variables X can lead to overfitting, especially when the latent dimension is moderate or when the dataset is small. Moreover, the standard formulation does not offer a principled way to quantify uncertainty in the latent representation. To address these issues, we turn to the Bayesian extension of the model.

Bayesian Gaussian Process Latent Variable Models (B-GPLVM) introduce a prior distribution over the latent variables and integrate them out through a variational approximation. In this setting, the optimisation proceeds by maximizing a variational lower bound on the marginal likelihood, and the incorporation of sparse inducing points ensures that the computational demands remain manageable. This approach yields a posterior distribution over the latent coordinates, improves generalisation, and provides uncertainty-aware latent embeddings that are suitable for the geometric analyses developed later in this chapter. The Bayesian formulation is particularly advantageous when working with limited data, noisy observations, or applications where explicit uncertainty quantification is essential.

The maximum likelihood estimation of the latent inputs $\mathcal{X}$ in GPLVM is known to be prone to overfitting, particularly when the dimensionality of the latent space is not small [27]. A Bayesian treatment of the latent inputs offers a way to mitigate this issue by introducing a prior distribution over $\mathcal{X}$. Under this formulation, the marginal likelihood becomes

$$p(S) = \int p(S \mid \mathcal{X})\, p(\mathcal{X})\, d\mathcal{X}.$$

However, this integral is not tractable in closed form, as the latent variables enter the

model through nonlinear operations involving the inverse of the covariance matrix.

The Bayesian Gaussian Process Latent Variable Model (B-GPLVM; [28]) addresses this difficulty by employing a variational inference framework. In this approach, the latent variables are integrated out variationally, yielding a computable lower bound on the marginal likelihood of the nonlinear latent variable model. Optimizing this variational bound provides a Bayesian training procedure that reduces the tendency to overfit and results in a more stable and uncertainty-aware latent representation.

The key ingredient that makes the variational Bayes treatment tractable is the use of a sparse GP formulation in which the GP prior on ϕ is augmented with a set of auxiliary variables. Specifically, following [29], we expand the conditional model by introducing m additional function evaluations of the latent mapping. Each such inducing point takes the form $u_i = \phi(x_{u,i}) \in \mathbb{R}^p$, and the collection of inducing variables is

$$\mathcal{U} = \{u_i^j \mid i = 1, \dots, m,\ j = 1, \dots, p\}, \qquad \mathcal{U} \in \mathbb{R}^{m \times p}.$$

These correspond to evaluations of the function at a set of pseudo-inputs

$$\mathcal{X}_u = \{x_{u,i}^j \mid i = 1, \dots, m,\ j = 1, \dots, q\}, \qquad \mathcal{X}_u \in \mathbb{R}^{m \times q},$$

which are treated as variational parameters.

Under this augmented representation, the joint probability model and its corresponding marginal likelihood take the form

$$\begin{aligned} p(S, \Phi, \mathcal{U}, \mathcal{X} \mid \beta) &= p(S \mid \Phi)\, p(\Phi \mid \mathcal{U}, \mathcal{X})\, p(\mathcal{U})\, p(\mathcal{X}) \\ &= \prod_{j=1}^{p} p(s^j \mid \phi^j)\, p(\phi^j \mid u^j, \mathcal{X})\, p(u^j)\, p(\mathcal{X}), \end{aligned} \tag{2.20}$$

$$p(S) = \iiint p(S, \Phi, \mathcal{U}, \mathcal{X})\, d\mathcal{U}\, d\Phi\, d\mathcal{X}. \tag{2.21}$$

The integral in 2.21 is not analytically tractable, but a variational approximation can be constructed by applying Jensen's inequality to $\log p(S)$ [27]. The resulting lower bound admits a closed-form expression, and its full derivation is provided in Eq C.10 in Appendix C.

For a point x_* in the latent space, the distribution of the Jacobian under B-GPLVM

retains the same structure as in Eq 2.13 and is given by

$$p(J \mid \mathcal{X}, \mathcal{U}, S) = \prod_{j=1}^{p} \mathcal{N}\big(\mu^j, \Sigma_j\big) \tag{2.22}$$

$$= \prod_{j=1}^{p} \mathcal{N}\Big(\partial K_{x_*}^{\top} K_{\mathcal{X}_u\mathcal{X}_u}^{-1} \mu_{\mathcal{U}}^j,\ \partial^2 K_{x_*x_*} - \partial K_{x_*}^{\top} \Lambda\, \partial K_{x_*}\Big), \tag{2.23}$$

$$\Lambda = K_{\mathcal{X}_u\mathcal{X}_u}^{-1} - K_{\mathcal{X}_u\mathcal{X}_u}^{-1} \Sigma_{\mathcal{U}} K_{\mathcal{X}_u\mathcal{X}_u}^{-1}, \tag{2.24}$$

and the predictive variance of the mapping at x_* becomes

$$\mathrm{Var}(\phi(x_*) \mid x_*) = K_{x_*x_*} - K_{x_*\mathcal{X}_u} \Lambda\, K_{\mathcal{X}_u x_*}. \tag{2.25}$$

Here $\mu_{\mathcal{U}}^j$ and $\Sigma_{\mathcal{U}}$ denote the mean and covariance of the variational posterior of the inducing variables. The expectation of the metric tensor follows the same form as in Eq 2.17, and the boundary $\partial\mathcal{M}$ of the learned manifold can be obtained by evaluating the mapping variance in Eq 2.25.

2.3 Variational Autoencoder

Autoencoders (AEs) learn deterministic mappings between data and a low-dimensional latent representation through an encoder–decoder architecture. However, AEs lack a probabilistic formulation of the latent space and provide no principled way to model data uncertainty. Variational Autoencoders (VAEs) extend this framework by introducing a generative latent-variable model together with variational inference.

VAEs are a class of latent-variable generative models that combine variational inference with deep neural networks [81]. They can be viewed as a probabilistic extension of standard autoencoders: instead of mapping an input deterministically to a single latent code, VAEs learn a distribution over latent variables that captures variability and uncertainty in the data. This probabilistic formulation leads to a smooth and structured latent space, which is useful for both generative modeling and downstream representation learning.

Model Formulation. The VAE assumes that each observation x is generated from an unobserved latent variable z through a prior $p(z)$ and a likelihood (decoder) $p_\theta(x \mid$

z). A common choice is a standard normal prior $p(z) = \mathcal{N}(0, I)$ and a neural network decoder that parameterises $p_\theta(x \mid z)$. Since the true posterior $p_\theta(z \mid x)$ is intractable in general, VAEs introduce an approximate posterior (encoder) $q_\phi(z \mid x)$, typically chosen as a diagonal Gaussian whose mean and variance are predicted by another neural network.

The learning objective follows directly from the variational inference framework established in classical work [79, 80]. Introducing the approximate posterior and applying Jensen's inequality yields the evidence lower bound (ELBO):

$$\log p_\theta(x) \;\geq\; \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] - \mathrm{KL}\big(q_\phi(z \mid x) \,\|\, p(z)\big), \tag{2.26}$$

where the reconstruction term encourages fidelity to the data under the generative model, and the KL divergence regularises the latent representation toward the prior. This decomposition later popularised in the context of deep generative models by Kingma and Welling [81]-captures the fundamental trade-off between reconstruction accuracy and latent regularity.

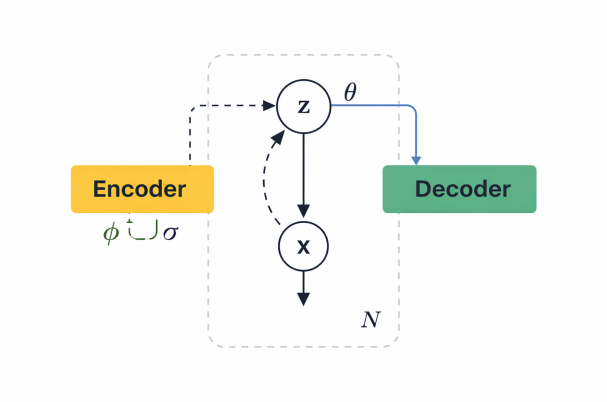


Figure 2.5: Schematic illustration of the variational autoencoder, showing the generative model $p_\theta(x \mid z)$ and the variational posterior $q_\phi(z \mid x)$. Adapted from [81].

Reparameterisation Trick. In order to optimise the ELBO with gradient-based methods, one needs to differentiate through samples drawn from $q_\phi(z \mid x)$. The reparameterisation trick addresses this by expressing z as a deterministic transformation of an auxiliary noise variable:

$$z = \mu_\phi(x) + \sigma_\phi(x)\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{2.27}$$

where $\mu_\phi(x)$ and $\sigma_\phi(x)$ are outputs of the encoder network. This formulation isolates the stochasticity in ϵ and makes the ELBO differentiable with respect to ϕ, allowing standard backpropagation to be used.

In practice, VAEs provide a continuous latent representation that supports interpolation, sampling, and regularised feature learning. Although they may suffer from issues

such as blurry reconstructions or posterior collapse in some settings, they remain a widely used and conceptually simple baseline in generative modeling. In this thesis, the VAE serves primarily as a reference point for later diffusion-based models and latent-variable methods, offering both a baseline probabilistic formulation and a conceptual bridge to more advanced architectures.

A schematic illustration of the encoder-decoder architecture and the associated latent-variable graphical model is shown in Figure 2.5, which helps to visualise the relationship between the generative model $p_\theta(x \mid z)$, the prior $p(z)$, and the variational posterior $q_\phi(z \mid x)$.

2.3.1 VAE Metric

In this thesis, we employ a pretrained VAE to map input images from the high-dimensional pixel space to a compact latent space that captures the essential semantic structure. The encoder, denoted by ϕ^{-1}, serves as a feature extractor, mapping an image z_i to a latent representation $x_i = \phi^{-1}(z_i)$, while the decoder ϕ reconstructs the image via $\hat{z}_i = \phi(x_i)$. This encoder–decoder architecture is kept fixed during the training of the diffusion model, providing a bidirectional mapping between the image and latent spaces and enabling efficient diffusion operations in the latent domain. In particular, the decoder defines a smooth mapping back to the original image space.

Following the Riemannian metric defined in Eq. (2.17), the metric tensor $\mathbf{g}$ is given by

$$\mathbf{g} = \mathbf{J}^\top \mathbf{J}, \tag{2.28}$$

where $\mathbf{J}$ denotes the Jacobian matrix, which can be interpreted as the partial derivative $\frac{\partial \phi(x_*)}{\partial x}$. In the VAE decoder, $\mathbf{J}$ can be computed via the chain rule by propagating derivatives through the network layers.

For a Gaussian decoder with diagonal covariance, it is natural to consider the diagonal immersion $x \mapsto (\mu_\theta(x), \sigma_\theta(x))$, which leads to the following VAE-based pull-back metric:

$$g(x) = \mathbf{J}_{\mu_\theta}(x)^\top \mathbf{J}_{\mu_\theta}(x) + \mathbf{J}_{\sigma_\theta}(x)^\top \mathbf{J}_{\sigma_\theta}(x). \tag{2.29}$$

Chapter 3

Diffusion Models Background

This chapter provides a systematic overview of diffusion-based generative modeling and establishes the theoretical foundations upon which the proposed geometry-aware diffusion framework is built. We first introduce the forward (noising) and reverse (denoising) stochastic processes and their connection to **Score-based Generative Models (SGMs)** through the score matching objective. This formulation enables generative modeling by learning the gradient of the data log-density and performing sampling via SDEs.

We then review the **Latent Diffusion Model (LDM)** framework, which improves computational efficiency by performing diffusion in a compact latent space rather than directly in the high-dimensional data space. Key components such as time-dependent score parameterisation, reverse-time SDE sampling, and classifier-free guidance are introduced, as they form the standard backbone of modern latent diffusion models.

Despite their empirical success, existing latent diffusion models rely on implicit Euclidean assumptions in latent space and do not explicitly account for the intrinsic geometry of the learned latent manifolds. This geometric mismatch can lead to distorted sampling dynamics, degraded global coherence, and reduced robustness in regions of high uncertainty. These limitations motivate the development of geometry-aware diffusion frameworks, which are the central focus of the subsequent chapters.

3.1 Score-Based Generative Models

Generative modeling aims to learn the underlying probability distribution of real data such that new samples can be drawn from it. Recent research suggests that, instead of directly modeling a normalised probability density, one can instead focus on learning the score function, i.e., the gradient of the log-density with respect to the input. This idea forms

the basis of Score-Based Generative Models (SGMs) [36], which also provide a unifying perspective for diffusion probabilistic models. This approach offers several advantages. First, it avoids the need to estimate the normalisation constant (partition function), which is often intractable in high-dimensional spaces. Second, learning the score function can be more stable and scalable, as it converts density estimation into a regression problem.

In this section, we first explain the definition and meaning of the score function, then introduce noise-perturbed score matching, and finally present the continuous-time formulation using SDEs. Discrete diffusion models, such as Denoising Diffusion Probabilistic Model (DDPM) [39], can be viewed as special cases within this general framework.

3.1.1 Problem Setting

Given a dataset $\{x_1, x_2, \ldots, x_N\}$ drawn independently from an unknown data distribution $p(x)$, the goal of generative modeling is to learn a model that can approximate $p(x)$ and generate new samples that resemble those in the dataset. This task is especially challenging in high-dimensional domains such as images, audio, and video, where the data distribution tends to be highly structured and concentrated around a complicated low-dimensional manifold embedded in a much larger ambient space. As a consequence, $p(x)$ assigns high density only to a very small fraction of the input space, while most regions have extremely low or negligible probability mass.

Many classical generative models attempt to directly model a normalised probability density. For instance, an energy-based model can be written as

$$p_\theta(x) = \frac{e^{-f_\theta(x)}}{Z_\theta}, \tag{3.1}$$

where $f_\theta(x)$ is a learnable energy function and $Z_\theta = \int e^{-f_\theta(x)}\, dx$ is the normalizing constant. However, evaluating or differentiating Z_θ is typically intractable in high dimensions, which makes maximum likelihood learning difficult in practice.

Other generative approaches, such as GANs [13], avoid the normalisation issue but introduce their own challenges. In particular, adversarial training can be unstable due to the minimax optimisation between the generator and discriminator, which may lead to oscillatory dynamics or failure to converge. In addition, GANs are prone to mode collapse, where the generator produces a limited diversity of samples and fails to capture the full data distribution [14].

Score-based generative models provide such a framework by shifting the modeling

target from the density $p(x)$ to its *score function*,

$$\nabla_x \log p(x).$$

This gradient describes how the log-density changes around x and, crucially, does not involve the intractable normalisation term since $\nabla_x \log Z_\theta = 0$. As discussed in the next section, the score function has a clear geometric interpretation: it points toward directions where the data density increases most rapidly, effectively capturing local structure of the data distribution. Once the score function is learned, it can be used within stochastic sampling procedures, such as Langevin dynamics or reverse-time diffusion, to gradually transform random noise into realistic samples.

In the following sections, we introduce the basic ingredients of score-based generative modeling. We begin by defining the score function and its learning objective, then extend the discussion to noise-perturbed score matching, and finally move toward continuous-time formulations that unify score-based models with diffusion probabilistic models.

3.1.2 Score Function and Score Matching

Score-based generative models build on the idea that instead of modeling a normalised probability density, one can learn its *score function*, defined as

$$s(x) = \nabla_x \log p(x). \tag{3.2}$$

Although this expression appears simple, the quantity it represents is fundamental. Since $\nabla_x \log p(x) = \frac{1}{p(x)} \nabla_x p(x)$, the score vector at x always points toward directions in which the data density increases most rapidly. In other words, it describes a vector field over the input space that locally indicates where the distribution becomes more likely. For points lying away from the data manifold, the magnitude of the score tends to be large, pulling them back toward the high-density region. This geometric interpretation explains why the score function is sufficient for guiding sampling processes such as Langevin dynamics.

To motivate score matching, recall that normalised density models

$$p_\theta(x) = \frac{e^{-f_\theta(x)}}{Z_\theta},$$

suffer from the difficulty that the normalizing constant Z_θ is typically intractable in high-dimensional settings. However, taking the gradient of the log-density removes Z_θ entirely, since it does not depend on x. Thus learning the score function bypasses the main obstacle

of maximum-likelihood training.

Formally, the goal is to approximate the true score $\nabla_x \log p(x)$ with a neural network $s_\theta(x)$. This is achieved by minimizing the *Fisher divergence*:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)} \left[\tfrac{1}{2} \left\| \nabla_x \log p(x) - s_\theta(x) \right\|_2^2 \right]. \tag{3.3}$$

The key advantage of this loss is that it does not require direct access to $\nabla_x \log p(x)$; instead, it admits reformulations that depend only on the data samples themselves, making score matching feasible in practice.

Once a score model is trained, new data samples can be synthesised using Langevin dynamics. The starting point is the continuous-time SDE

$$dx_t = \frac{1}{2} \nabla_x \log p(x_t)\, dt + d\mathbf{w}_t, \tag{3.4}$$

where $\mathbf{w}_t$ denotes standard Brownian motion. This SDE describes a process that moves toward regions of higher data density while injecting noise to ensure sufficient exploration. Discretizing the SDE with a step size $\epsilon > 0$ gives the update rule

$$x_{t+1} = x_t + \frac{\epsilon}{2} \nabla_x \log p(x_t) + \sqrt{\epsilon}\, \mathbf{z}_t, \qquad \mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I}), \tag{3.5}$$

where ϵ controls the magnitude of both the deterministic movement and the injected noise. In practice, the true score is replaced by the learned estimator s_θ, yielding

$$x_{t+1} = x_t + \frac{\epsilon}{2} s_\theta(x_t) + \sqrt{\epsilon}\, \mathbf{z}_t. \tag{3.6}$$

Starting from random noise, this iterative refinement gradually transforms an initial point into a sample consistent with the data distribution.

The formulation above describes score learning on clean data. However, estimating score functions from sharply concentrated real-world distributions can be unstable, which motivates noise-perturbed and time-dependent extensions discussed in the following sections.

3.1.3 Noise-Perturbed Score Matching

Classical score matching assumes that the data density $p(\mathbf{x})$ is well-behaved over the entire ambient space. However, real-world data often concentrate around a low-dimensional manifold, making $\nabla_x \log p(x)$ poorly defined or unstable in low-density regions. To alleviate this issue, [42] introduced *denoising score matching* (DSM), which learns the score of

a *smoothed* version of the data distribution.

DSM alleviates this issue by learning the score of a *single* noise-perturbed distribution. Given a corruption process

$$x = y + \sigma z, \qquad y \sim p(x),\ z \sim \mathcal{N}(0, \mathbf{I}),$$

the perturbed density is the Gaussian convolution

$$p_\sigma(x) = (p * \mathcal{N}(0, \sigma^2 \mathbf{I}))(x),$$

and the DSM objective is

$$\mathcal{L}_{\mathrm{DSM}}(\theta) = \mathbb{E}_{p(y)} \mathbb{E}_{\mathcal{N}(x; y, \sigma^2 \mathbf{I})} \left[\left\| s_\theta(x) + \frac{x - y}{\sigma^2} \right\|_2^2 \right]. \tag{3.7}$$

To understand the appearance of the term $(x - y)/\sigma^2$, note that the corruption distribution is Gaussian:

$$p(x \mid y) = \mathcal{N}(x; y, \sigma^2 \mathbf{I}),$$

whose log-density satisfies

$$\nabla_x \log p(x \mid y) = -\frac{x - y}{\sigma^2}. \tag{3.8}$$

Thus DSM implicitly trains the network to match

$$s_\theta(x) \approx \nabla_x \log p_\sigma(x),$$

i.e., the score of a smoothed distribution that has full support and is much easier to estimate reliably.

While DSM significantly improves stability, using only a single noise level σ restricts the range of perturbations the model learns to handle. To provide richer training signals and better conditioning across different noise regimes, [36] extended DSM to multiple Gaussian noise levels

$$\sigma_1 < \sigma_2 < \cdots < \sigma_L.$$

For each scale σ_i, we define a noise-perturbed distribution by convolving the original data distribution $p(x)$ with a Gaussian kernel:

$$p_{\sigma_i}(x) = \int p(y)\, \mathcal{N}(x; y, \sigma_i^2 \mathbf{I})\, dy.$$

Sampling from p_{σ_i} is straightforward: we draw a clean data point $y \sim p(x)$ and add Gaussian noise,

$$x = y + \sigma_i z, \qquad z \sim \mathcal{N}(0, \mathbf{I}).$$

As shown in Figure 3.1, adding Gaussian noise with increasing standard deviations

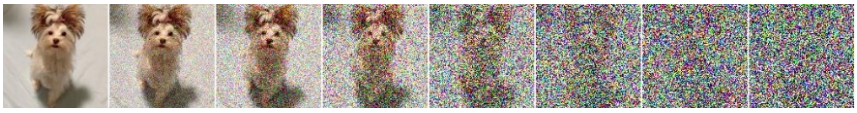


Figure 3.1: Noise perturbation process: clean data is corrupted by Gaussian noise of increasing variance levels $\sigma_1 < \sigma_2 < \cdots < \sigma_L$, producing a sequence of progressively smoother distributions.

smooths the data distribution and stabilises score estimation. At small noise levels, p_{σ_i} remains close to the original data distribution; at larger noise levels, the distribution becomes more diffuse and easier to model. Figure 3.1 illustrates how increasing Gaussian noise progressively smooths the data.

A noise-conditional score network $s_\theta(x, i)$ is then trained to approximate the score function of each perturbed distribution:

$$s_\theta(x, i) \approx \nabla_x \log p_{\sigma_i}(x).$$

Instead of learning a single score function, the model learns to predict scores across a range of noise levels. This multi-scale formulation stabilises training by providing reliable learning signals even in regions where the original distribution is difficult to estimate.

The overall training objective for noise-perturbed score matching is then written as

$$\mathcal{L}(\theta) = \sum_{i=1}^{L} \lambda(i)\, \mathbb{E}_{p_{\sigma_i}(x)} \left[\left\| \nabla_x \log p_{\sigma_i}(x) - s_\theta(x, i) \right\|_2^2 \right], \tag{3.9}$$

where $\lambda(i) > 0$ is a weighting coefficient that balances the contributions of different noise levels. A common and effective choice is $\lambda(i) = \sigma_i^2$, which compensates for the fact that the score magnitude typically decreases as the noise scale increases. This weighting helps stabilise optimisation, ensuring that no particular noise level dominates the objective.

By learning score functions across multiple noise scales, the model receives informative gradients even in low-density regions of the original distribution. The smoothed, noise-perturbed distributions make score estimation better conditioned and lead to improved sample quality and diversity.

Figure 3.2 illustrates how increasing the noise level smooths the data distribution and produces a more regular score field. Large noise scales yield stable, easy-to-learn scores,

while smaller scales capture finer details of the data manifold, motivating the multi-scale design of annealed denoising score matching.

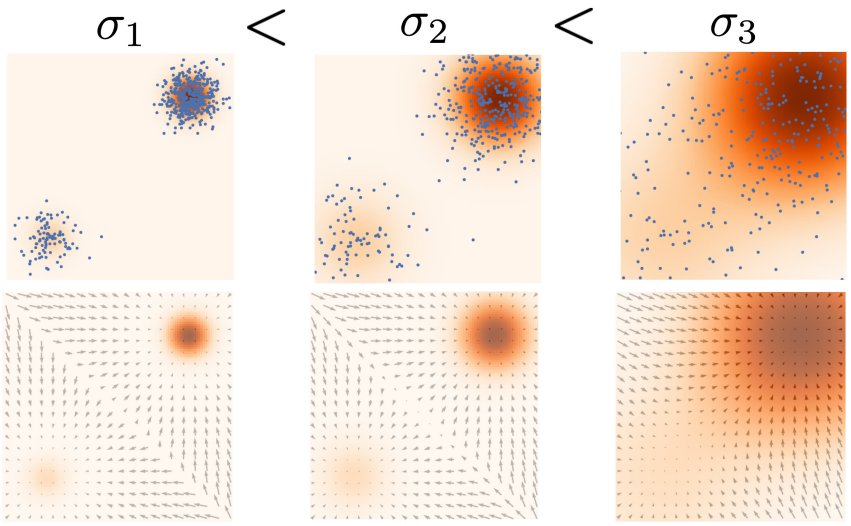


Figure 3.2: Effect of Gaussian noise perturbation on data samples (top row) and the corresponding score fields (bottom row) under increasing noise scales $\sigma_1 < \sigma_2 < \sigma_3$. Higher noise levels produce smoother distributions and more stable score estimates, motivating annealed (multi-scale) denoising score matching. Adapted from visualisations by Song et al. [36, 37].

To further generalise this idea, the discrete noise levels $\{\sigma_i\}$ can be replaced by a continuous noise schedule parameterised by a time variable $t \in [0, T]$. In this case, the score network becomes time-dependent, $s_\theta(x, t)$, and is trained to approximate $\nabla_x \log p_t(x)$ for continuously perturbed data. This continuous-time formulation forms the basis of time-dependent score-based generative models described in the next subsection.

3.1.4 Time-Dependent Score-Based Model

The discrete noise scales introduced in the previous section can be generalised into a continuous perturbation process. Instead of defining a finite sequence $\{\sigma_i\}_{i=1}^{L}$, we introduce a continuous time variable $t \in [0, T]$ and construct a family of perturbed distributions $\{p_t(x)\}_{t\in[0,T]}$. Here, $t = 0$ corresponds to the original data distribution $p_0(x) = p(x)$, while $t = T$ represents a highly corrupted distribution that approaches a simple prior, such as a Gaussian. This continuous formulation avoids the need for manually choosing a discrete set of noise levels.

A common way to define such a continuous perturbation process is through a SDE of the form

$$dx = \mathbf{f}(x, t)\, dt + g(t)\, d\mathbf{w}, \tag{3.10}$$

where $\mathbf{f}$ is the drift term, $g(t)$ is the diffusion coefficient, and $\mathbf{w}$ denotes standard Brownian motion. The marginal distribution of the solution $x(t)$ at time t is denoted by $p_t(x)$. As t increases, the injected noise gradually dominates, causing p_t to approach a tractable prior.

To illustrate, consider the SDE

$$dx = \sigma^t \, d\mathbf{w}, \quad t \in [0, 1], \tag{3.11}$$

where $\sigma > 1$ is a fixed hyperparameter. In this case, the perturbation process is purely diffusive and injects noise with exponentially growing variance. The transition kernel from time 0 to t has a closed form:

$$p_{0t}(x(t) \mid x(0)) = \mathcal{N}\left(x(t); x(0), \frac{1}{2\log\sigma}\left(\sigma^{2t} - 1\right)\mathbf{I}\right). \tag{3.12}$$

Thus, perturbations interpolate smoothly between the data distribution at $t = 0$ and a near-isotropic Gaussian distribution at $t = 1$. When σ is large, the marginal distribution at $t = 1$ becomes approximately

$$p_1(x) \approx \mathcal{N}\left(x; \mathbf{0}, \frac{1}{2\log\sigma}(\sigma^2 - 1)\,\mathbf{I}\right),$$

which is independent of the data distribution and easy to sample from.

Time-dependent score function. To model the evolving perturbed distributions $\{p_t\}$, we train a time-dependent score network $s_\theta(x, t)$ to approximate their score functions:

$$s_\theta(x, t) \approx \nabla_x \log p_t(x).$$

This leads to a continuous-time extension of score matching:

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{p_t(x)}\left[\lambda(t)\left\|s_\theta(x, t) - \nabla_x \log p_t(x)\right\|_2^2\right]. \tag{3.13}$$

Here $\lambda(t) > 0$ is a weighting function used to balance the relative contributions of different noise levels. Because the scale of $\nabla_x \log p_t(x)$ may vary significantly across t, an appropriate choice of $\lambda(t)$ stabilises training. In practice, $\lambda(t)$ is often chosen to be inversely proportional to the expected squared gradient magnitude,

$$\lambda(t) \propto \frac{1}{\mathbb{E}\left[\|\nabla_{x(t)} \log p_t(x(t))\|_2^2\right]}, \tag{3.14}$$

so that no particular time interval dominates the optimisation.

For the exponential SDE above, the natural weighting function is

$$\lambda(t) = \frac{1}{2\log\sigma}(\sigma^{2t} - 1), \tag{3.15}$$

which matches the variance of the perturbation kernel p_{0t}. This ensures that all noise levels contribute proportionally to the overall learning signal.

As illustrated in Figure 3.3, the forward SDE gradually perturbs data into noise, while a corresponding reverse-time SDE can be used to denoise and recover the data distribution. This continuous-time framework unifies score-based generative models and diffusion probabilistic models.

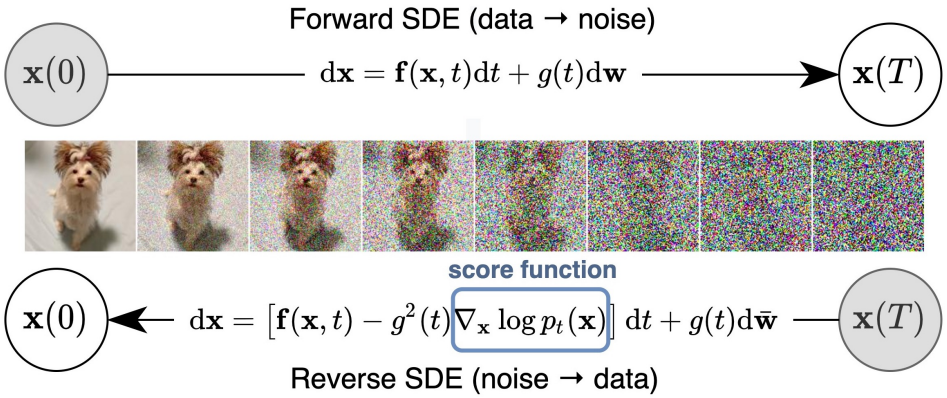


Figure 3.3: Forward perturbation SDE and reverse-time denoising SDE. This continuous framework unifies score-based generative models and diffusion models. (Figure adapted from [37], CC-BY 2021)

3.1.5 Reverse SDE and Sampling

Having defined a forward perturbation process that gradually transforms the data distribution into a simple prior, we now describe how to reverse this process to generate new samples. The key result, originating from stochastic process theory, is that under mild regularity conditions, the forward SDE

$$dx = \mathbf{f}(x, t)\, dt + g(t)\, d\mathbf{w}$$

admits a corresponding *reverse-time SDE* that describes the evolution of $x(t)$ when time is run backward from T to 0. This reverse-time SDE involves the score function of the perturbed distribution $p_t(x)$.

The reverse-time dynamics corresponding to the forward SDE are given by

$$dx = \left[\mathbf{f}(x,t) - g(t)^2 \nabla_x \log p_t(x)\right] dt + g(t)\, d\bar{\mathbf{w}}, \tag{3.16}$$

where $\bar{\mathbf{w}}$ denotes Brownian motion in reverse time. This expression shows how the score function plays a crucial role: while $g(t)\, d\mathbf{w}$ injects noise in the forward process, the term $-g(t)^2 \nabla_x \log p_t(x)$ appears in the reverse process to remove this noise, effectively denoising the signal as t decreases.

In practice, the true score $\nabla_x \log p_t(x)$ is unknown and is replaced by the learned score model $s_\theta(x,t)$, yielding the approximate reverse SDE

$$dx = \left[\mathbf{f}(x,t) - g(t)^2 s_\theta(x,t)\right] dt + g(t)\, d\bar{\mathbf{w}}, \tag{3.17}$$

where $\bar{\mathbf{w}}$ denotes Brownian motion when time runs backward. The additional term involving the score function exactly compensates, in expectation, for the noise introduced by the forward SDE.

We have $\mathbf{f}(x,t) = 0$ and $g(t) = \sigma^t$, so the reverse SDE becomes

$$dx = -\sigma^{2t} \nabla_x \log p_t(x)\, dt + \sigma^t d\bar{\mathbf{w}}. \tag{3.18}$$

Replacing the true score with the learned model s_θ, we obtain the practical sampling equation:

$$dx = -\sigma^{2t} s_\theta(x,t)\, dt + \sigma^t d\bar{\mathbf{w}}. \tag{3.19}$$

To generate new samples:

1. Draw $x(T)$ from the prior distribution p_T, which is approximately Gaussian.
2. Discretise the reverse SDE (e.g., Euler–Maruyama).
3. Evolve $x(t)$ backward from $t = T$ to $t = 0$.

The resulting sample $x(0)$ approximates a draw from the data distribution p_0.

Role of $\lambda(t)$ in sampling. Although $\lambda(t)$ is important in the training objective—balancing contributions from different noise levels—it does not appear in the reverse SDE. Sampling depends only on the forward SDE $(\mathbf{f}, g)$ and the learned score model $s_\theta(x,t)$; $\lambda(t)$ affects how well the score is learned, but not the sampling dynamics.

Special cases. Different choices of $(\mathbf{f}, g)$ recover various diffusion-type models. For example, the Variance Exploding (VE) and Variance Preserving (VP) SDEs proposed in [37] correspond to two commonly used perturbation processes. Discrete diffusion models such as DDPM [39] can be interpreted as particular discretisations of VP-type SDEs. Thus, time-dependent score-based models and diffusion probabilistic models can be understood within a single unified continuous-time framework.

3.2 Sliced Score Matching

In many score-based generative modeling settings, the perturbed distribution $p_t(x)$ may not admit a tractable density or a closed-form transition kernel. This situation commonly arises when the perturbation process is defined through a general SDE, where evaluating or differentiating $\log p_t(x)$ is infeasible. Consequently, classical score matching objectives, which rely on the score $\nabla_x \log p_t(x)$, cannot be applied directly.

Sliced Score Matching (SSM) [41] addresses this challenge by introducing a density-free approach for estimating $\nabla_x \log p_t(x)$. Instead of requiring explicit access to the probability density, SSM leverages random projections to approximate the Fisher divergence. The key idea is to match directional derivatives of the score function along random directions $v \sim \mathcal{N}(0, \mathbf{I})$, reducing the need for evaluating the full score of $p_t(\mathbf{x})$.

Formally, the SSM training objective for a time-dependent score model $s_\theta(x, t)$ is given by

$$\theta = \arg\min_\theta \; \mathbb{E}_t\Big[\lambda(t)\, \mathbb{E}_{x(0)}\, \mathbb{E}_{x(t)}\, \mathbb{E}_{v\sim\mathcal{N}(0,\mathbf{I})}\big[\tfrac{1}{2}\|s_\theta(x(t),t)\|_2^2 + v^\top \nabla_{\mathbf{x}} s_\theta(x(t),t)v\big]\Big], \tag{3.20}$$

where $x(t)$ is samplled from the forward perturbation process, v is a random projection direction, and $\lambda(t)$ is the temporal weighting coefficient used previously in continuous-time score matching.

This objective bypasses the need for evaluating $p_t(x)$ or its gradient. The first term penalises the magnitude of the score estimate, while the second term matches the expected directional derivatives of the score along random Gaussian directions. Together, they provide an unbiased estimate of the Fisher divergence without requiring $\nabla_x \log p_t(x)$ in closed form.

While SSM is broadly applicable and particularly useful for perturbation processes with intractable transition densities, the reliance on random projections can introduce high variance in gradient estimates, especially in high-dimensional spaces. In practice, variance-reduction strategies—such as using multiple projection directions per batch—

are commonly employed to improve training stability.

Despite these challenges, SSM remains a foundational tool for density-free score estimation and serves as a stepping stone for score estimation techniques in more complex geometries. In Section 4.3, we build upon this framework when extending score-based methods to manifold-structured latent spaces.

3.3 Latent Diffusion Model

Building upon the continuous-time score-based framework introduced in the previous section, we now turn to the Latent Diffusion Model (LDM) [35], which shifts the diffusion process from the pixel space to a compact, perceptually aligned latent space. This design substantially reduces computational cost while retaining high generative quality.

Conventional diffusion models operate directly on high-dimensional images, requiring significant memory and computation, especially for high-resolution settings. Moreover, the score network must simultaneously model fine-grained textures and global semantics, leading to inefficient use of model capacity [39, 40].

The LDM addresses these limitations by first mapping images to a low-dimensional latent space using a pretrained autoencoder. Diffusion and denoising are then performed entirely within this latent representation, allowing the score model to focus on semantically meaningful structure rather than raw pixel statistics. The decoder subsequently maps denoised latents back into the image domain.

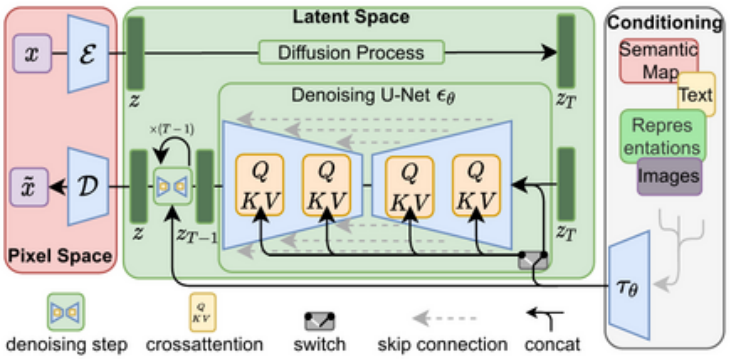


Figure 3.4: Overview of the Latent Diffusion Model (LDM). Reproduced from [35]. A pretrained autoencoder encodes the input image into a compact latent space. A diffusion model is trained on these latents, and the denoised latent representation is decoded back into pixel space to obtain the final output.

As illustrated in Figure 3.4, the latent diffusion process is parameterised by a U-Net backbone that optionally incorporates self and cross attention layers. These attention

mechanisms enable conditional generation (e.g., text-to-image synthesis) but are not the primary focus of this dissertation and will therefore not be discussed in detail.

3.3.1 Diffusion in Latent Space

Let $\tilde{\mathbf{x}}_0$ denote the latent representation of an image produced by a pretrained encoder. The diffusion process in LDM follows the same continuous-time formulation as presented previously, but applied directly in the latent domain. That is, the forward perturbation process is given by an SDE of the general form

$$d\tilde{x}(t) = g(t)\, d\mathbf{w}(t), \tag{3.21}$$

where $g(t)$ follows a prescribed noise schedule. A common choice is the variance-exploding (VE) schedule $g(t) = \sigma^t$, which progressively injects larger noise as t increases and drives $\tilde{x}(t)$ toward a Gaussian distribution.

The corresponding reverse-time SDE—which performs denoising—is therefore

$$d\tilde{x} = -g(t)^2\, s_\theta(\tilde{x}, t)\, dt + g(t)\, d\bar{\mathbf{w}}(t), \tag{3.22}$$

with s_θ representing the score network defined in latent space. For the VE schedule, this specialises to

$$d\tilde{x} = -\sigma^{2t} s_\theta(\tilde{x}, t)\, dt + \sigma^t\, d\bar{\mathbf{w}}(t). \tag{3.23}$$

Compared with pixel-space diffusion, the dimensionality of $\tilde{x}$ is orders of magnitude smaller, meaning both the forward corruption and the reverse denoising processes are substantially more efficient.

3.3.2 Training Objective

The score network is trained using the same denoising score matching principle introduced earlier, but evaluated entirely in latent space. Let $\tilde{x}_t$ denote the noisy latent produced by the forward SDE.

A practical training objective used in LDMs is given by

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,T),\, x_0\sim p_{\text{data}},\, z\sim\mathcal{N}(0,I)} \left[\left\| s_\theta(\tilde{x}_t, t)\, \tau(t) + z \right\|_2^2 \right],$$

where $\tilde{x}_t$ is obtained by perturbing x_0 according to the forward SDE. Thus, the score model learns to predict the noise direction associated with latent perturbations.

This objective is mathematically equivalent to the DDPM/SDE training losses described previously, but shifted into the latent domain via the autoencoder.

3.3.3 Sampling

Once trained, sampling proceeds by integrating the reverse SDE from $t = 1$ to $t = 0$, starting from a Gaussian latent variable. Using Euler–Maruyama discretisation, the update rule becomes

$$\tilde{x}_{t-\Delta t} = \tilde{x}_t + \sigma^{2t} s_\theta(\tilde{x}_t, t)\,\Delta t + \sigma^t \sqrt{\Delta t}\,\bar{z}, \quad \bar{z} \sim \mathcal{N}(0, \mathbf{I}). \tag{3.24}$$

The resulting latent $\tilde{x}_0$ is then decoded into pixel space to produce the final sample.

In summary, the latent diffusion model leverages the expressive power of score-based generative modeling while greatly reducing computational cost through latent-space training. This design has proven highly effective for high-resolution image synthesis and forms the basis of the methods developed in the following chapters.

3.4 Classifier-free guidance

To further enhance conditional controllability and improve semantic alignment in generative sampling, we adopt the **classifier-free guidance (CFG)** strategy [45]. This approach jointly trains a conditional and an unconditional diffusion model within a single framework and combines their score estimates during sampling. Unlike *classifier guidance* [40], which relies on an external discriminative model to steer generation, classifier-free guidance performs conditioning intrinsically, eliminating the need for an auxiliary classifier while maintaining strong controllability and high sample fidelity.

Formally, we define an unconditional denoising diffusion model $p_t(x)$ parameterised through a score estimator $s_\theta(x, t)$, and a conditional diffusion model $p_t(x \mid c)$ parameterised through $s_\theta(x, t, c)$, where c denotes the conditioning variable (e.g., class label, text embedding, or other side information). Both score networks share the same neural architecture, typically a U-Net backbone, differing only in whether the conditional information c is provided. For the unconditional model, the conditioning input is replaced by a null token:

$$s_\theta(x, t) = s_\theta(x, t, c = \emptyset). \tag{3.25}$$

During training, we jointly optimise both the unconditional and conditional score estimators within a unified framework. Specifically, for each training sample $(x, c) \sim p(x, c)$, the conditioning variable c is randomly set to $\emptyset$ with a probability p_{uncond}. This stochastic

masking mechanism allows the model to learn both conditional and unconditional denoising objectives simultaneously:

$$c \leftarrow \begin{cases} c, & \text{with probability } 1 - p_{\text{uncond}}, \\ \emptyset, & \text{with probability } p_{\text{uncond}}. \end{cases} \tag{3.26}$$

The hyperparameter p_{uncond} thus controls the trade-off between conditional and unconditional modeling capacity. When $p_{\text{uncond}} \to 0$, the training effectively reduces to a purely conditional diffusion model, whereas higher values allocate part of the model's capacity to unconditional generation.

[45] systematically investigated the impact of varying p_{uncond}, and reported that only a small portion of the model's capacity needs to be devoted to the unconditional task in order to achieve effective classifier-free guidance. In practice, values of $p_{\text{uncond}} \in \{0.1, 0.2\}$ were found to be sufficient. This finding parallels the observation of [40], who noted that even low-capacity classifiers can yield strong guidance effects for classifier-based methods. Both results suggest that a limited degree of unconditional modeling or auxiliary discrimination is sufficient to significantly improve conditional sample quality.

During the sampling (backward) process, classifier-free guidance is applied by forming a linear combination of the conditional and unconditional score estimates:

$$\tilde{s}_\theta(x, t, c) = (1 + w)s_\theta(x, t, c) - w\, s_\theta(x, t). \tag{3.27}$$

Intuitively, this linear combination amplifies the contribution of the conditional score while partially subtracting the unconditional component. As a result, the generative trajectory is steered toward samples that better align with the conditioning variable c, without requiring explicit gradients from an auxiliary classifier.

Here, $w \geq 0$ denotes the *guidance strength*. Increasing w typically enhances the conditioning fidelity (e.g., sharper class or text alignment) but may also reduce sample diversity. In practice, moderate values of $w \in [1, 5]$ often yield a favorable trade-off between diversity and quality.

Algorithm 2 and Algorithm 3 summarise the overall training and sampling procedures for classifier-free guidance in conditional diffusion models.

Algorithm 2 Training with classifier-free guidance (CFG)

Input: dataset $p(x, c)$; unconditional prob. p_{uncond}
Initialise: parameters θ
Repeat until convergence:
$(x, c) \sim p(x, c)$ { sample data and condition }
with prob. p_{uncond}, set $c \leftarrow \emptyset$ { unconditional branch }
sample forward-diffusion paths
update θ by minimizing Eq. (4.6)

Algorithm 3 Conditional sampling with classifier-free guidance (CFG)

Input: guidance w; condition c
Output: sample x
initialise $x_T \sim \mathcal{N}(0, I)$
for $t = n_{t_b}, \ldots, 1$ { n_{t_b} reverse steps, step size Δt_b } **do**
$\tilde{s}_\theta(x_t, t, c) \leftarrow (1 + w)\, s_\theta(x_t, t, c) - w\, s_\theta(x_t, t)$
sample x_{t-1} from the reverse process using $\tilde{s}_\theta$
end for
Return: x_0

Chapter 4

Intrinsic Hybrid Latent Diffusion Model

Building upon the foundations of Riemannian geometry and stochastic manifold learning in Chapter 2 and score-based diffusion models in Chapter 3, this chapter presents the proposed Intrinsic Hybrid Latent Diffusion Model (ILDM). While LDMs [35] enhance efficiency by operating in a compressed latent space, they implicitly assume a Euclidean structure that ignores the intrinsic geometry of the learned data manifold. This simplification often results in geometric inconsistency and limited generalisation, particularly in regions poorly supported by training data.

To address this limitation, ILDM introduces an adaptive hybrid diffusion process that integrates manifold-aware (Riemannian) and Euclidean dynamics. The model follows the intrinsic geometry of the manifold in well-characterised regions, while switching to Euclidean diffusion in areas of high uncertainty, where the underlying manifold structure may be unreliable. This hybrid design ensures both preservation of the underlying geometry and numerical stability during generative sampling.

Furthermore, ILDM incorporates a unified score estimation framework and extends classifier-free guidance (CFG) to the manifold setting, enabling controllable generation aligned with the latent geometry. Together, these components establish a principled framework that unifies geometric learning and score-based generation within a single diffusion model.

The remainder of this chapter is organised as follows. Section 4.2 describes the hybrid forward diffusion process and its switching criterion. Section 4.3 presents the time-dependent score estimation framework. Section 4.4 details the hybrid backward process, and Section 4.5 discusses controllable generation with classifier-free guidance.

4.1 Intrinsic Hybrid Latent Diffusion Model

LDMs [35] construct a compact representation of high-dimensional data using techniques such as VAEs [47] or GPLVMs [29, 48]. Diffusion processes are then performed in this latent space for efficient generative modeling (see Section 3.3 for background).

However, the latent space is typically treated as Euclidean, without explicitly incorporating the geometric structure induced by the underlying data manifold. As a result, the diffusion dynamics may become inconsistent with the intrinsic geometry, particularly in regions of high curvature or limited data support.

To address this, we build upon the learned mapping ϕ and its associated Riemannian metric g (see Section 2.2.2). Our goal is to construct stochastic dynamics that (i) respect the intrinsic manifold structure in well-supported regions, and (ii) remain stable under uncertainty where the learned geometry may be unreliable.

In standard score-based generative modeling, data are progressively perturbed across multiple noise scales, and the evolution of perturbed distributions is governed by *stochastic differential equations* (SDEs) in Euclidean space. Here, we extend this framework to the manifold setting by defining diffusion processes that evolve on the learned latent manifold endowed with the Riemannian metric g. However, the reliability of g depends critically on the accuracy of the learned mapping ϕ, since g is derived from the local Jacobian as $g = J_\phi^\top J_\phi$. In regions poorly supported by the training data, where the decoder must extrapolate, the estimated metric may become unreliable—exhibiting spurious curvature or degenerate eigenvalues. Such distortions can compromise geodesic distances and cause unstable diffusion dynamics.

To mitigate these issues, we introduce a *hybrid diffusion approach*. In regions of low mapping uncertainty—where the learned manifold is well characterised—the stochastic process follows the manifold's intrinsic geometry. Conversely, in high-uncertainty regions, where the estimated metric is unreliable and manifold information is less meaningful, the process reverts to standard Euclidean diffusion dynamics. This adaptive scheme maintains geometric consistency in well-learned regions while ensuring stability and robustness elsewhere, effectively balancing preservation of the manifold geometry and generalisation.

In the following sections, we formalise the forward and backward diffusion processes within this hybrid framework and describe the corresponding *score estimation strategy*, accounting for the absence of closed-form transition densities on general manifolds. An overview of the proposed framework is illustrated in Figure 4.1.

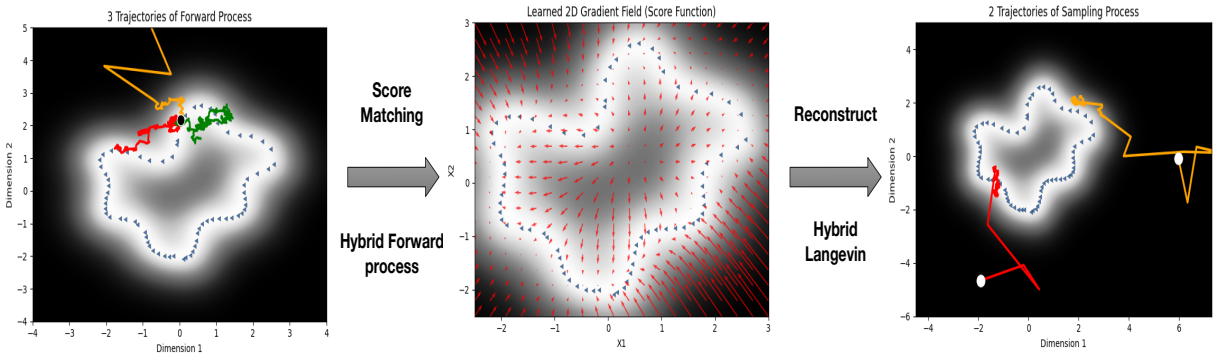


Figure 4.1: Score-based generative model on the data noise manifold. The blue triangles in each sub-figure represent the data points in the latent space. The left sub-figure shows three Brownian motion (BM) paths in the latent space. Starting from the black circle, the red and green paths follow Riemannian BM, while the yellow path initially follows Riemannian BM but switches to Euclidean BM when it enters the high uncertainty (dark) region. Once the BM paths are simulated, the vector field of score function ($\nabla_x \log p(x(t))$) is learned as described in Section 3.1.2. The gradients at some grid points are shown in middle sub-figure, and they clearly point toward the data points (blue triangles). The learned scores are then used to construct the backward process. The right panel shows two backward paths, which start from white circles and converge toward the data distribution.

4.2 Forward Process

We build upon Brownian motion on manifolds (Section 2.1.2) and diffusion processes described in Chapter 3. We define the forward process $\{x(t)\}_{t=0}^{T}$ on the learned latent manifold, indexed by a continuous time variable $t \in [0, T]$. Let $p_t(x)$ denote the probability density of $x(t)$. The initial state follows $x(0) \sim p_0$, where p_0 represents underlying data distribution that generates the observations. The terminal state follows $x(T) \sim p_T$, where p_T serves as the prior distribution for sample generation. The transition density from $x(t_i)$ to $x(t_j)$ is denoted by $p_{t_i t_j}(x(t_j) \,|\, x(t_i))$. In the proposed hybrid formulation, the forward process alternates between two diffusion mechanisms depending on the local uncertainty of the learned mapping ϕ. In regions of low uncertainty, we simulate *Riemannian Brownian motion* (BM) that evolves intrinsically along the manifold geometry. Conversely, in high-uncertainty regions, the process transitions to *Euclidean Brownian motion*, corresponding to a variance-exploding (VE) SDE as introduced in [41]. This design allows the forward dynamics to respect geometric structure where the manifold is reliable while maintaining stability in regions where the learned geometry becomes uncertain.

Given the estimated metric g introduced in Section 2.2.2, the BM on a Riemannian

manifold in local coordinates can be expressed as a system of SDEs in the Itô form [17,18]:

$$dx^i(t) = \frac{1}{2} G^{-1/2} \sum_{j=1}^{q} \frac{\partial}{\partial x^j} \left({g^{-1}}_{ij} G^{1/2} \right) dt + \left(g^{-1/2} db(t) \right)_i , \tag{4.1}$$

where G denotes the determinant of g, $b(t)$ is an independent Euclidean Brownian motion, and i indexes the latent space coordinates.

The discretised form of these SDEs can be obtained via the Euler–Maruyama method [49, 50]:

$$\begin{aligned} x^i(t) &= x^i(t - \Delta t) + \frac{1}{2} \sum_{j=1}^{q} \left(-g^{-1} \frac{\partial g}{\partial x^j} g^{-1} \right)_{ij} \Delta t + \frac{1}{4} \sum_{j=1}^{q} (g^{-1})_{ij} \mathrm{tr} \left(g^{-1} \frac{\partial g}{\partial x^j} \right) \Delta t + \left(g^{-1/2} db(t) \right)_i \\ &= \mu\big(x^i(t - \Delta t), \Delta t\big) + \left(\sqrt{\Delta t}\, g^{-1/2} z_t \right)_i , \end{aligned} \tag{4.2}$$

where Δt denotes the diffusion time step, $\mathrm{tr}(\cdot)$ is the trace operator, z_t is a q-dimensional standard Gaussian random vector, and $\mu(\cdot)$ represents the deterministic drift term composed of the first three components in Eq. (4.2). Both the drift and diffusion terms explicitly depend on the metric $g(x)$, which varies spatially in the latent space. Consequently, the Riemannian BM is reliable only in regions where the estimated metric is stable, i.e., where the mapping ϕ exhibits low predictive uncertainty.

Hybrid Switching. The reliability of the Riemannian metric is quantified by the predictive variance $\sigma^2(x)$ of the mapping $\phi(x)$, which provides a natural measure of local uncertainty. While other uncertainty measures could be used, predictive variance offers a convenient and well-defined criterion in our setting. As a point in the latent space moves away from the training data, the variance increases and eventually plateaus at the maximum value $\sigma^2_{\max}$. For Gaussian process (GP) decoders, this upper bound corresponds to the model's prior variance. To determine whether the forward process should follow manifold-based or Euclidean dynamics, we introduce a switching threshold determined as a fraction of this maximum variance:

$$\sigma^2_{\text{thresh}} = \alpha\, \sigma^2_{\max}, \quad \alpha \in (0, 1).$$

When $\sigma^2(x) < \sigma^2_{\text{thresh}}$, the process follows Riemannian BM on the learned manifold; otherwise, it transitions to Euclidean BM, equivalent to a variance-exploding (VE) SDE [41]. This adaptive rule allows the model to leverage the geometric structure where ϕ is confident, while permitting stable and flexible diffusion behavior in high-uncertainty regions. The hybrid forward process does not admit a closed-form transition density $p_{0t}(x_t|x_0)$ due to the spatially varying metric and stochastic switching mechanism. Nevertheless, the

hybrid SDE can be simulated numerically using schemes such as Euler–Maruyama.

Figure 4.1 (left) illustrates three representative hybrid forward trajectories initialised from the same starting point (black circle). Two trajectories (red and green) remain as Riemannian BM throughout, while the third (yellow) switches to Euclidean dynamics upon entering a high-uncertainty region. The complete simulation procedure is summarised in Algorithm 4 in section 4.3.

Algorithm 4 Forward Diffusion Process on the Learned Manifold (Hybrid Formulation)

Require: Learned mapping $\phi(\cdot)$ and corresponding metric $g(x)$ estimated from the point cloud $\mathcal{S} = \{s_i \mid i = 1, \ldots, n_d\}$;
n_f: number of starting points;
n_{BM}: number of Brownian motion trajectories per starting point;
n_{t_f}: number of diffusion steps;
Δt: integration step size;
σ^2_{thresh}: variance-based switching threshold.

Ensure: Simulated hybrid diffusion trajectories $\{x_{i,j}(t)\}$.

for $i = 1$ **to** n_f **do**
  Randomly select or initialise a starting point $x_i(0)$ from the data distribution p_0.
  for $j = 1$ **to** n_{BM} **do**
    Initialise trajectory $x_{i,j}(0) \leftarrow x_i(0)$.
    for $l = 1$ **to** n_{t_f} **do**
      Compute predictive variance $\sigma^2(x_{i,j}(l-1)) \leftarrow \mathrm{Var}[\phi(x_{i,j}(l-1))]$.
      if $\sigma^2(x_{i,j}(l-1)) < \sigma^2_{\text{thresh}}$ **then**
        (Manifold regime) Simulate a Riemannian Brownian motion step using the local metric $g(x_{i,j}(l-1))$:

$$q(x_{i,j}(l) \mid x_{i,j}(l-1)) = \mathcal{N}\big(x_{i,j}(l) \mid \mu(x_{i,j}(l-1), \Delta t)\,, \Delta t\, g^{-1}(x_{i,j}(l-1))\big)\,,$$

        where $\mu(\cdot)$ denotes the drift term defined in Eq. (4.2).
      else
        (Euclidean regime) Switch to a Euclidean Brownian motion step with isotropic covariance:

$$q(x_{i,j}(l) \mid x_{i,j}(l-1)) = \mathcal{N}(x_{i,j}(l) \mid x_{i,j}(l-1), \Delta t\, I)\,.$$

      end if
      Sample $x_{i,j}(l) \sim q(x_{i,j}(l) \mid x_{i,j}(l-1))$.
    end for
  end for
end for
return $\{x_{i,j}(t)\}$ as the set of simulated hybrid forward diffusion trajectories.

To further understand the role of the switching threshold parameter α in controlling the transition between Riemannian and Euclidean diffusion, we conduct an ablation study on the same dataset used in our main experiments. Specifically, we empirically examine $\alpha \in \{0.3, 0.4, 0.5\}$, corresponding to thresholds at 30%–50% of the maximum predictive

variance $\sigma^2_{\max}$. Across all tested settings, the proposed intrinsic latent diffusion model (ILDM) consistently achieves lower Fréchet Inception Distance (FID) scores compared to the standard LDM, indicating enhanced sample quality and improved alignment with the data manifold.

The parameter α effectively balances geometric fidelity and numerical stability. Smaller α values lead to earlier switching into the Euclidean regime, prioritizing robustness at the expense of geometric detail, whereas larger α values preserve Riemannian dynamics over broader regions, emphasizing manifold consistency but risking instability where the metric is unreliable. In practice, α can be treated as a tunable hyperparameter and selected through cross-validation or empirical calibration based on the data distribution and the uncertainty characteristics of the learned mapping ϕ. While this work specifically considers BM in $\mathbb{R}^q$ for high uncertainty regions, our framework is flexible and could incorporate alternative SDEs in high uncertainty region, such as those described in [41].

4.3 Estimating Scores via Time-Dependent Score Matching

Following the time-dependent score matching framework introduced in Section 3.1.2, we learn a time-dependent score function $\nabla_x \log p_t(x)$, where $p_t(x)$ denotes the marginal density of the hybrid forward process. In conventional latent diffusion models (LDMs), where diffusion evolves in a Euclidean latent space with a known closed-form transition density, *denoising score matching* (DSM) can be directly applied to train the score network using analytically tractable perturbation distributions [36, 42]. However, in ILDM, the forward process is governed by a hybrid stochastic differential equation. Although the drift and diffusion coefficients are analytically defined, the corresponding transition density $p_{0t}(x_t|x_0)$ does not admit a closed-form expression, preventing the direct application of standard DSM.

To address this challenge, we propose two alternative strategies based on simulated forward trajectories: (i) a *sliced score matching* objective (Section 3.2) adapted from [36], and (ii) a novel *Approximate Denoising Score Matching* objective developed in this work. Both methods rely on samples generated by simulating the hybrid forward SDE (Algorithm 4) to estimate the score network $s_\theta(x, t)$.

Sliced Score Matching (SSM). SSM provides a density-free alternative for score estimation by projecting the score gradient onto random directions. Following [41], the

training objective is defined as:

$$\theta = \arg\min_{\theta} \mathbb{E}_t \left\{ \lambda(t) \mathbb{E}_{x(0)} \mathbb{E}_{x(t)} \mathbb{E}_{v \sim p_v} \left[\frac{1}{2} \| s_\theta(x(t), t) \|_2^2 + v^\top \nabla_x s_\theta(x(t), t) v \right] \right\}, \tag{4.3}$$

where $v \sim p_v$ is a random direction vector drawn from a standard multivariate Gaussian distribution, and $\lambda(t) : [0, T] \to \mathbb{R}^+$ is a time-dependent weighting function. For each $x_0 \sim p_0$, the hybrid SDE is simulated forward to time t to obtain samples $x(t) \sim p_t$. The weighting function $\lambda(t)$ is set proportional to the empirical variance of x_t, following the noise-aware weighting strategy in [36]. The score network $s_\theta(x, t)$ is implemented as a convolutional U-Net. While SSM avoids dependence on explicit transition densities and is broadly applicable, it may converge slowly in high-dimensional settings due to the stochastic nature of random directional projections, which can introduce high-variance gradient estimation.

Approximate Denoising Score Matching (Approximate DSM). To address these limitations, we develop a more efficient and stable alternative, termed Approximate DSM. The central idea is to retain the denoising principle of standard DSM while relaxing the requirement of a tractable transition density $p_{t0}(x_t|x_0)$. Specifically, we approximate the conditional transition distribution by a Gaussian form whose parameters are estimated empirically from simulated trajectories of the forward SDEs:

$$p_{0t}(x_t \mid x_0) \approx \mathcal{N}(\tilde{\mu}_t, \tilde{\tau}_t), \quad \tilde{\mu}_t = \hat{\mu}_t, \quad \tilde{\tau}_t = \hat{\tau}_t, \tag{4.4}$$

where $\hat{\mu}_t$ and $\hat{\tau}_t$ denote the empirical mean and variance estimated from simulated trajectories of the forward SDE. Under this Gaussian approximation, the conditional score can be expressed as:

$$\nabla_x \log p_{0t}(x_t|x_0) \approx \frac{x_t - \tilde{\mu}_t}{\tilde{\tau}_t}. \tag{4.5}$$

Accordingly, we define the approximate DSM loss as:

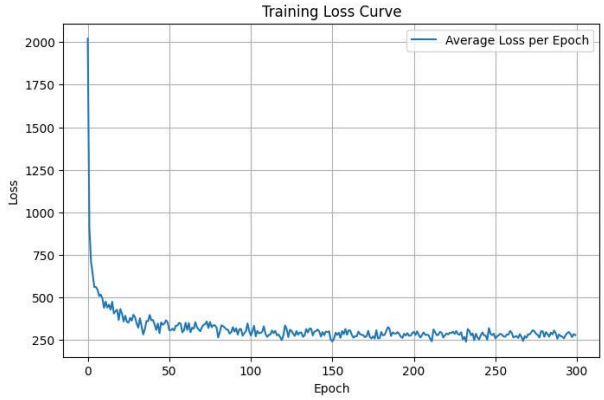
$$\theta = \arg\min_{\theta} \mathbb{E}_t \left\{ \lambda(t) \, \mathbb{E}_{x(0)} \, \mathbb{E}_{x(t)|x(0)} \left[\left\| s_\theta(x_t, t) \cdot \tilde{\tau}_t - \frac{x_t - \tilde{\mu}_t}{\tilde{\tau}_t} \right\|^2 \right] \right\}, \tag{4.6}$$

where $\lambda(t)$ is a time-dependent weighting function balancing gradient magnitudes across diffusion stages.

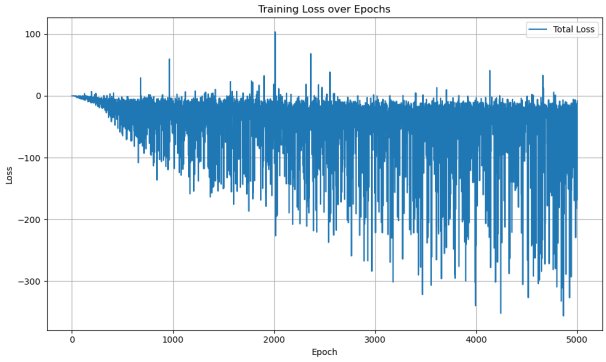
This formulation bypasses random projections and enables more stable and efficient optimisation. Although it assumes a Gaussian form for the conditional distribution, our empirical results demonstrate that the resulting score estimation closely matches that obtained from SSM. In both methods, evaluating the score network $s_\theta(x_t, t)$ at grid points in

latent space yields a vector field, which can be interpreted as an estimate of the gradient of the log-density. Throughout, we use the terms score function and vector field interchangeably. Additional comparisons of vector field visualisations and training stability are provided below.

Figure 4.2 presents the learning curve for the two score estimation methods. The vertical axis represents the loss, while the horizontal axis corresponds to training epochs. The upper panel shows that optimisation with the approximate DSM objective converges smoothly within roughly 150–200 epochs, indicating stable learning dynamics. In contrast, the lower panel demonstrates that SSM optimisation exhibits large fluctuations and slower convergence, with high variance persisting even after 5000 epochs. These results confirm that the approximate DSM objective provides a more reliable and computationally efficient alternative for score estimation in ILDM.



(a) ADSM loss.

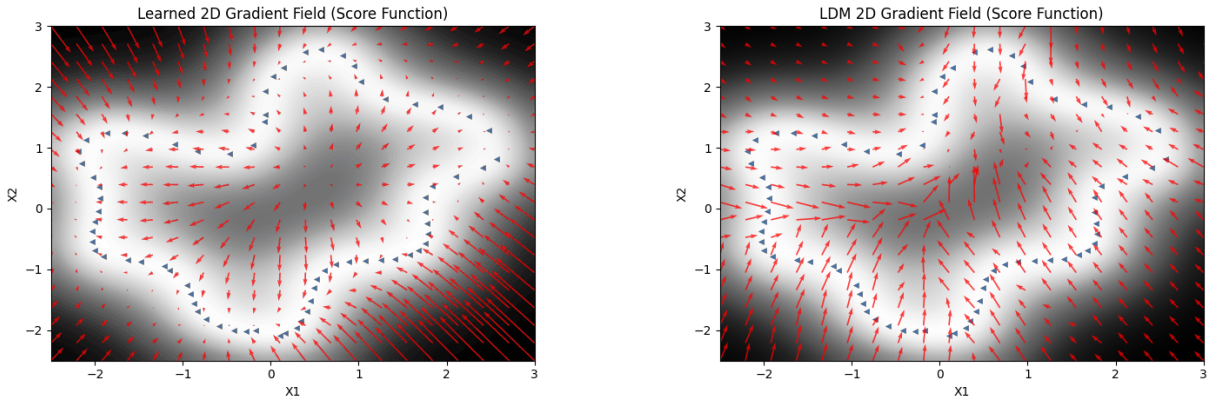


(b) Sliced score matching (SSM) loss.

Figure 4.2: Optimisation losses of different score matching objectives across training epochs. The ADSM objective stabilises after approximately 200 epochs, while the SSM objective continues to exhibit high variance even after 5000 epochs.

We further compare the estimated score fields obtained from ILDM (trained via Approximate DSM) and from the standard LDM on the COIL image dataset. Details of the dataset are provided in Section 5.2. Figure 4.3 visualises the learned vector fields in the

latent space. In ILDM (left panel), the estimated vectors consistently point toward the data samples (blue triangles), faithfully capturing the local geometry of the data manifold. In contrast, the LDM-based vector field (right panel) predominantly points toward the center of the latent space, where no data exist, suggesting a failure to recover the true geometric structure of the data distribution.



(a) ILDM vector field

(b) LDM vector field

Figure 4.3: Comparison of vector field estimates in the COIL latent space. (a) ILDM produces geometry-consistent score vectors aligned with the data manifold, while (b) LDM yields center-biased vectors that deviate from the intrinsic data structure.

4.4 Backward Process

Once the time-dependent score network $s_\theta(x_t, t)$ has been trained, we construct the backward stochastic process that reverses the hybrid forward diffusion and enables generative sampling from the target data distribution p_0. Analogous to the forward process, the backward dynamics must adapt to the spatially varying uncertainty of the learned mapping ϕ. To this end, we formulate the reverse-time dynamics as a *hybrid of Euclidean and Riemannian Langevin processes* [43, 44]. In regions where the learned metric is reliable, the backward dynamics evolve intrinsically along the manifold geometry; in high-uncertainty regions, the process switches to Euclidean Langevin updates to maintain numerical stability.

We begin sampling from terminal states $x(T) \sim p_T$, and simulate the process backward in time until time 0, which samples from x_0. In regions where the predictive uncertainty of ϕ is high, the backward dynamics turn to the Euclidean Langevin diffusion

$$dx(t) = -\frac{1}{2}\nabla_x \log p_t(x(t))\, dt + db(t), \tag{4.7}$$

where $b(t)$ denotes a standard Euclidean Brownian motion and dt represents an infinites-

imal negative timestep. Replacing the intractable score function $\nabla_x \log p_t(x)$ with the learned estimator $s_\theta(x_t, t)$, the SDE can be simulated via a numerical solver, such as Euler–Maruyama. The corresponding discretisation for the i-th coordinate is

$$\begin{aligned} x^i_{t-\Delta t} &= x^i_t - \frac{1}{2} s_\theta(x_t, t)_i \, \Delta t + \sqrt{\Delta t}\, z^i_t, \\ &= \mu(x^i_t, \Delta t) + \left(\sqrt{\Delta t}\, z_t\right)_i, \end{aligned} \tag{4.8}$$

where $z_t \sim \mathcal{N}(0, I_q)$ is a q-dimensional Gaussian random vector.

In contrast, when the local uncertainty of ϕ is low, the backward dynamics follow the Riemannian Langevin diffusion on the learned manifold endowed with metric tensor $g(x)$:

$$dx(t) = -\frac{1}{2} g^{-1}(x_t) \nabla_x \log p_t(x_t)\, dt + d\tilde{\mathcal{B}}(t), \tag{4.9}$$

where $\tilde{\mathcal{B}}(t)$ is Brownian motion defined intrinsically on the manifold. Substituting $s_\theta(x_t, t)$ for the true score yields the coordinate expansion

$$dx^i(t) = -\left(\frac{1}{2} g^{-1} s_\theta(x_t, t)\, dt\right)_i + \frac{1}{2} G^{-1/2} \sum_{j=1}^{q} \frac{\partial}{\partial x^j} \left(g^{-1}_{ij} G^{1/2}\right) dt + (g^{-1/2} db(t))_i, \tag{4.10}$$

where $G = \det(g)$. The corresponding discrete update is

$$x^i_{t-\Delta t} = \tilde{\mu}(x_t, \Delta t) + \left(\sqrt{\Delta t}\, g^{-1/2} z_t\right)_i, \quad z_t \sim \mathcal{N}(0, I_q), \tag{4.11}$$

where $\tilde{\mu}(x_t, \Delta t)$ denotes the deterministic Riemannian drift terms.

Hybrid Switching Strategy. To achieve robust sampling across regions with varying mapping uncertainty, we employ a **hybrid backward process** that adaptively switches between the Euclidean and Riemannian updates. At each time step, the local uncertainty of the mapping function ϕ is estimated. If the local variance $\mathrm{Var}[\phi(x_t)] > \sigma^2_{\text{thresh}}$, the Euclidean update (4.7) is applied; otherwise, the Riemannian update (4.11) is used. The variance threshold σ^2_{thresh} is defined analogously to Section 4.2.

At each reverse-time step, we compute the local predictive variance $\sigma^2(x_t)$ of the mapping $\phi(x)$. Following the same thresholding scheme introduced in Section 4.2, the Riemannian Langevin update (4.11) is applied when $\sigma^2(x_t) < \sigma^2_{\text{thresh}}$, and the Euclidean update (4.7) is used otherwise. This adaptive mechanism ensures that the backward process preserves the manifold geometry when it is trustworthy, while maintaining stable sampling behavior in regions where the metric is unreliable.

Figure 4.1 (right panel) illustrates example trajectories of the backward hybrid pro-

cess. Two paths are initialised from points in high-uncertainty regions (dark), far from the data distribution. The trajectories initially follow the Euclidean Langevin diffusion and move with large steps. As they enter the low uncertainty region (white), the dynamics switch to Riemmanian Langevin updates and converge towards the data distribution. The complete backward sampling procedure is presented in Algorithm 5.

Algorithm 5 Hybrid Backward Langevin Process on an Unknown Manifold

Require: n_b: Number of backward points , n_{La}: number of trajectories, n_{t_b}: number of backward steps, step size Δt_b, variance threshold σ^2_{thresh}, metric $\mathcal{G}(x)$, score network $s_\theta(x,t)$, terminal states $x_T \sim p_T$

Ensure: Reconstructed samples $\tilde{x} \sim p_0$

Initialise $x_{i,j}(n_{t_b}) \leftarrow x_T$ for $i = 1, \dots, n_b$, $j = 1, \dots, n_{\text{La}}$
for $i = 1$ **to** n_b **do**
  for $j = 1$ **to** n_{La} **do**
    for $l = n_{t_b}$ **to** 1 **do**
      Compute local variance $\sigma^2_{i,j}(l) = \text{Var}[\phi(x_{i,j}(l))]$
      if $\sigma^2_{i,j}(l) < \sigma^2_{\text{thresh}}$ **then**
        {Riemannian update}
        $\tilde{\mu} \leftarrow x_{i,j}(l) - \frac{1}{2}\,\mathcal{G}^{-1}(x_{i,j}(l))\, s_\theta(x_{i,j}(l), t_l)\, \Delta t_b$
        $x_{i,j}(l-1) \sim \mathcal{N}(\tilde{\mu},\, \Delta t_b\, \mathcal{G}^{-1}(x_{i,j}(l)))$
      else
        {Euclidean update}
        $\mu = x_{i,j}(l) - \frac{1}{2} s_\theta(x_{i,j}(l), t_l)\Delta t_b$
        $x_{i,j}(l-1) \sim \mathcal{N}(\mu, \Delta t_b I)$
      end if
    end for
  end for
end for
return $\tilde{x} = \{x_{i,j}(0)\}$

4.5 Controllable Generation with Classifier-Free Guidance

To enable controllable and class-conditional generation within the ILDM framework, we incorporate *classifier-free guidance (CFG)* [45]. Rather than revisiting the standard formulation (see Section 3.4), we focus on how CFG integrates with the hybrid geometric structure of ILDM.

Concretely, we assume a score network that supports both conditional and unconditional evaluations, denoted by $s_\theta(x, t, c)$ and $s_\theta(x, t)$, respectively. During sampling, these

are combined using the standard CFG interpolation:

$$\tilde{s}_\theta(x, t, c) = (1 + w)s_\theta(x, t, c) - ws_\theta(x, t), \tag{4.12}$$

where $w \geq 0$ controls the strength of conditioning.

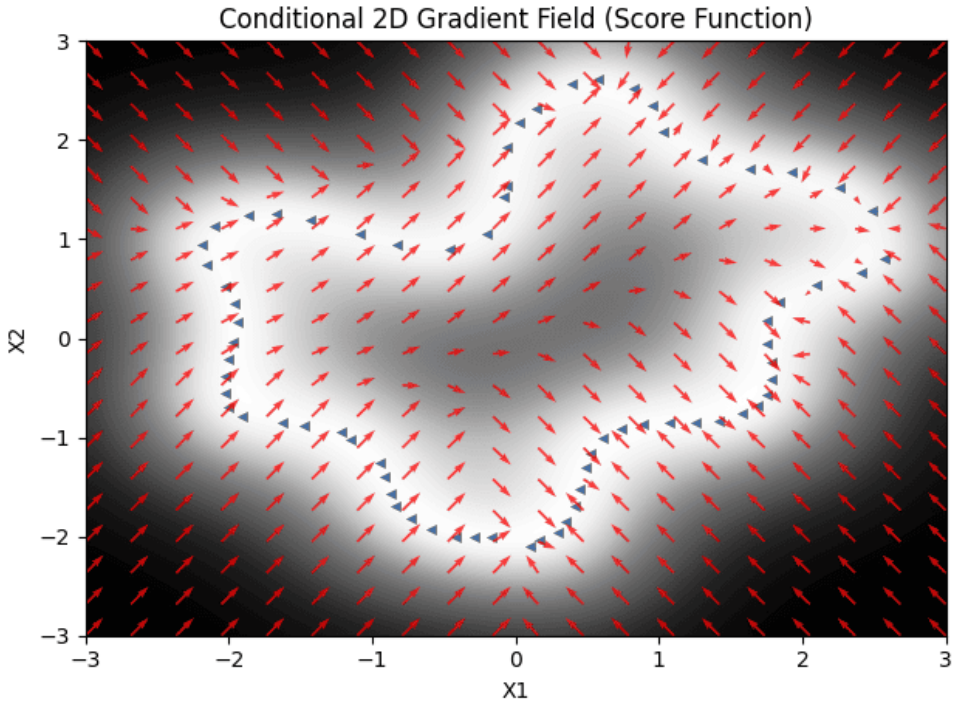
Unlike conventional latent diffusion models, ILDM operates under a hybrid backward SDE that couples Euclidean and Riemannian updates. In this setting, the guided score $\tilde{s}_\theta(x, t, c)$ is directly substituted into the backward dynamics described in Section 4.4. This design ensures that the guidance signal is applied consistently across both components of the flow.

A key advantage of this formulation is that conditioning remains aligned with the underlying manifold geometry. Because the score field itself is learned on the manifold-aware latent space, the guided vector field $\tilde{s}_\theta(x, t, c)$ naturally respects geometric constraints, avoiding the distortions or mode collapse often observed when guidance is applied in purely Euclidean latent spaces.

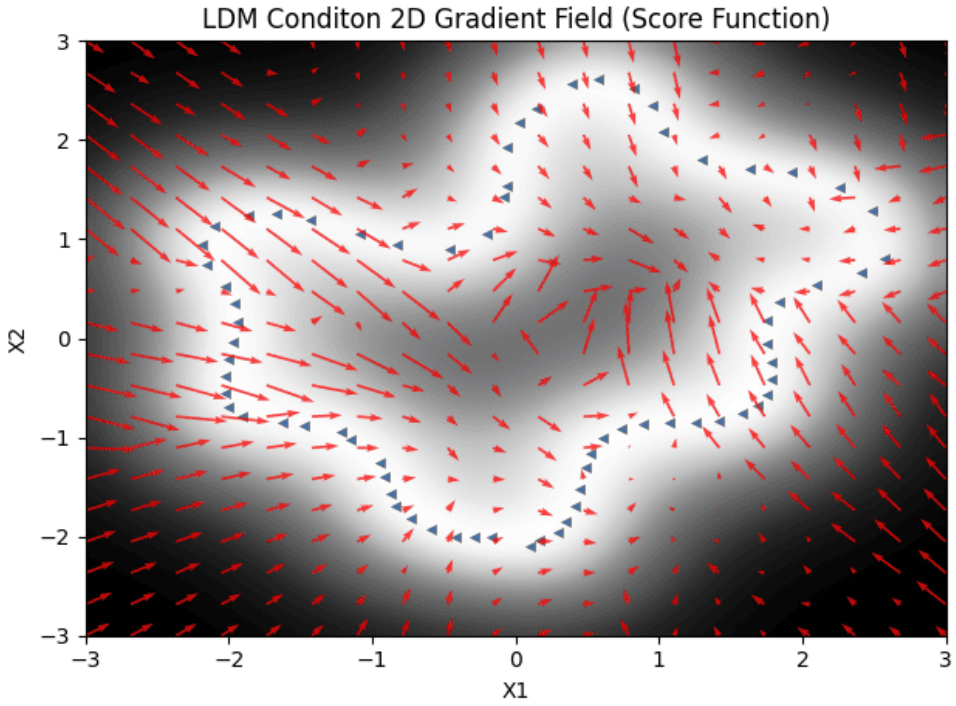
Figure 4.4 illustrates the resulting conditional score fields on the COIL latent space. Under the 'front-view' condition, ILDM produces vector fields that consistently point toward the corresponding semantic region, indicating that the guidance signal is coherently integrated with the data manifold. In contrast, the LDM baseline exhibits noisy and misaligned directions, suggesting a weaker coupling between conditioning and latent geometry.

In practice, we use $p_{\text{uncond}} \in \{0.1, 0.2\}$ during training and set the guidance weight to $w = 4$ during sampling, which provides a good balance between conditional fidelity and sample diversity.

Overall, CFG within ILDM provides a simple yet effective mechanism for controllable generation, while preserving the geometric structure of the latent space. This integration is crucial for achieving both high-quality samples and semantically meaningful trajectories in the diffusion process.



(a) ILDM conditional vector field



(b) LDM conditional vector field

Figure 4.4: Conditional vector fields in the COIL latent space under the front-view condition. Blue triangles denote latent data points from the conditional subset and red arrows denote the estimated score vectors.

Chapter 5

Diffusion Models Experiments

To evaluate the effectiveness of the proposed Intrinsic Latent Diffusion Model (ILDM)(Chapter 4), we conduct experiments on three benchmark datasets: the "Lucky cat" object from COIL-100 [51], a subset of MNIST [52], and cardiac MRI scans [53, 54]. These datasets differ significantly in structural characteristics and data dimensionality, providing a diverse testbed to examine ILDM's capability to adapt to varying underlying geometries. We compare ILDM with both standard diffusion models (DM) and LDM, using a combination of qualitative reconstructions and quantitative evaluations. For quantitative assessment, we adopt the Fréchet Inception Distance (FID) and Learned Perceptual Image Patch Similarity (LPIPS) metrics [55, 56] to measure generation quality and perceptual similarity.

5.1 Evaluation Metrics

To quantitatively assess the perceptual quality and realism of the generated images, we employ two widely adopted metrics in generative model evaluation: FID [55] and the LPIPS [56].

Fréchet Inception Distance (FID). FID measures how close the generated images are to real ones by comparing their feature distributions in the latent space of a pretrained Inception-V3 network [55]. For both real and generated images, we extract high-level semantic features from the final pooling (*pool3*) layer, obtaining two sets of feature vectors: $\mathcal{X}_r = \{x_r^{(i)}\}$ for real images and $\mathcal{X}_g = \{x_g^{(i)}\}$ for generated images. From each set, we compute an empirical mean and covariance, where μ_r and Σ_r denote the mean and covariance of real-image features, and μ_g and Σ_g denote those of the generated-image features. The

FID score is then given by

$$\text{FID} = \|\mu_r - \mu_g\|_2^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\right), \tag{5.1}$$

where $\text{Tr}(\cdot)$ denotes the trace operator. A lower FID indicates that the generated images are more similar to real ones in terms of both diversity and visual fidelity.

Learned Perceptual Image Patch Similarity (LPIPS). While FID measures the global distributional distance, LPIPS focuses on perceptual similarity between image pairs at the feature level. It computes the distance between deep feature representations extracted from a pretrained network (e.g., VGG or AlexNet), capturing human perceptual judgments more effectively than pixel-wise metrics such as PSNR or SSIM. Given two images x and y, LPIPS is defined as:

$$\text{LPIPS}(x, y) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|w_l \odot (f_l^x(h, w) - f_l^y(h, w))\|_2^2, \tag{5.2}$$

where f_l^x and f_l^y denote the deep feature maps of images x and y extracted at layer l, representing spatially structured activations encoding mid-level perceptual cues. Here, w_l are learned channel-wise weights, (h, w) index spatial locations, and H and W denote the height and width of the feature maps, respectively. Lower LPIPS values correspond to higher perceptual similarity. Following the formulation introduced by Zhang *et al.* [56], LPIPS has become a widely adopted perceptual metric for evaluating generative models.

Together, FID and LPIPS provide complementary insights: FID evaluates overall realism and diversity across the dataset, while LPIPS captures perceptual consistency and visual quality at the image level. For instance, images with realistic local textures but distorted global structure may exhibit low LPIPS yet high FID, whereas overly smooth images that match the overall data distribution may achieve good FID but worse LPIPS.

5.2 Coil-Images

We first evaluate the proposed ILDM using approximate denoising score matching on the Lucky Cat object from the COIL-100 dataset [51]. COIL-100 (Columbia Object Image Library) is a widely used benchmark for generative modeling and manifold learning, consisting of color images of 100 household objects acquired under controlled conditions. Each object is placed on a motorised turntable and photographed as it is rotated, resulting in a sequence of views that vary smoothly with the rotation angle while other factors (camera setup and illumination) are kept approximately fixed. In addition, COIL images are cap-

tured against a largely uniform background, which reduces irrelevant appearance variation and makes viewpoint the dominant source of change.

In our experiments we focus on the 'Lucky Cat' object subset, which contains 72 RGB images of a small porcelain figurine at a resolution of 128×128 pixels. The images are recorded at approximately 5° increments, spanning a full 360° rotation. This construction induces strong correlations between nearby frames and yields a dataset whose intrinsic degrees of freedom are much lower than the ambient pixel dimensionality. Intuitively, as the object rotates, the appearance evolves continuously and can be parameterised largely by the viewing angle (with minor additional variability arising from shading and self-occlusion). Consequently, the data are expected to concentrate near a low-dimensional nonlinear manifold embedded in the high-dimensional image space. This makes the Lucky Cat sequence a convenient testbed for assessing whether ILDM can exploit intrinsic geometry in latent space to produce coherent samples that respect smooth viewpoint transitions and preserve consistent global structure across generated views. Fig. 5.1 shows representative input images of the Lucky Cat object under several viewing angles, illustrating the smooth appearance variations induced by object rotation.

Figure 5.1: Coil Image Example images: showing the Lucky Cat object under 10 different viewing angles from the COIL dataset.

To determine the intrinsic dimensionality of the COIL dataset, we apply ARD to quantify the relative importance of each input dimension [78]. As shown in Fig. 5.2, the first two input dimensions exhibit significantly higher relevance scores compared to the remaining ones, indicating that most of the useful information is concentrated in these leading dimensions. Based on this analysis, the input data can be effectively reduced to a lower-dimensional representation without substantial loss of information, thereby improving computational efficiency and model generalisation.

A GPLVM (shown in section 2.2.1) is pretrained to embed the images to a two-dimensional latent space (shown in Figure 5.3) which defines the Riemannian metric used for diffusion.

Based on this latent parameterisation, we construct a time-conditioned score network to model the diffusion dynamics on the learned manifold. Table 5.1 summarises the architecture of the resulting ScoreNet, which adopts a U-Net–like encoder–decoder structure and incorporates temporal embeddings at each layer to enable time-dependent feature extraction and reconstruction.

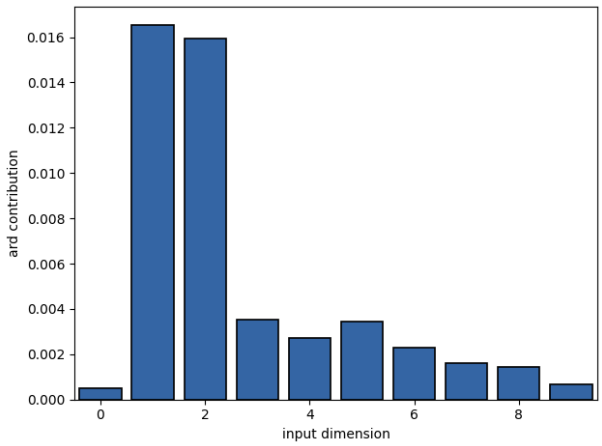


Figure 5.2: ARD (Automatic Relevance Determination) analysis on the COIL dataset. The bar heights indicate the relevance contribution of each input dimension, computed from the inverse squared length-scales of the ARD kernel. The first two dimensions show dominant importance, while the remaining contribute negligibly, suggesting that the effective latent dimensionality can be reduced to two in this setting.

Table 5.1: Architecture of the proposed training net for 1D CNN. The network follows a U-Net-like encoder–decoder design with time-step conditioning at every layer.

Module	Input Shape	Operation	Output Shape	Description
Input	(B, 1, L)	–	(B, 1, L)	1D input signal.
Time Embedding	(B, 1)	Fourier projection + Linear(embed_dim) + SiLU	(B, 256)	Temporal embedding vector.
Encoder Block 1	(B, 1, L)	Conv1D(1→32, k=3, p=1) + GN(4) + SiLU	(B, 32, L)	First encoder layer with temporal conditioning.
Encoder Block 2	(B, 32, L)	Conv1D(32→128, k=3, s=2, p=1) + GN(32) + SiLU	(B, 128, L/2)	Downsample and integrate time embedding.
Encoder Block 3	(B, 128, L/2)	Conv1D(128→256, k=3, s=2, p=1) + GN(32) + SiLU	(B, 256, L/4)	Deep encoder feature extraction.
Bottleneck	(B, 256, L/4)	Identity connection (optional)	(B, 256, L/4)	Latent representation.
Decoder Block 3	(B, 256, L/4)	ConvT1D(256→128, k=3, s=2, p=1) + GN(32) + SiLU	(B, 128, L/2)	Upsample features.
Decoder Block 2	(B, 128+128, L/2)	ConvT1D(256→32, k=3, s=2, p=1) + GN(32) + SiLU	(B, 32, L)	Combine with skip connections.
Decoder Block 1	(B, 32+32, L)	ConvT1D(64→1, k=3, p=1)	(B, 1, L)	Reconstruct denoised 1D signal.
Normalisation	(B, 1, L)	Divide by σ_t^2	(B, 1, L)	Scale output by noise variance.

To gain further insight into how the learned score networks behave in latent space, we visualise the corresponding vector fields induced by the LDM and ILDM models. These vector fields characterise the gradient directions of the log-density and therefore determine the stochastic dynamics that guide the sampling trajectories during the reverse diffusion process. Figure 5.4 shows the vector field learned by the LDM model after 10,000 training epochs and $\sigma = 11$. The gradient vectors mostly point toward the center region, forming an overall convergent pattern. This indicates that the model's learned score field tends to pull samples toward a central area, rather than toward the true latent points. As a result, many arrows fail to align with the true latent data structure. Figure 5.5 shows the vector field learned by the ILDM model, trained using 100 Brownian motion paths with 100 time steps and step size 0.5. Here, the vectors are much better aligned with the latent data points, pointing directly toward the manifold (i.e., the true latent points). This demonstrates that ILDM learns a more accurate gradient direction in latent space, capturing the geometry of the data distribution more faithfully.

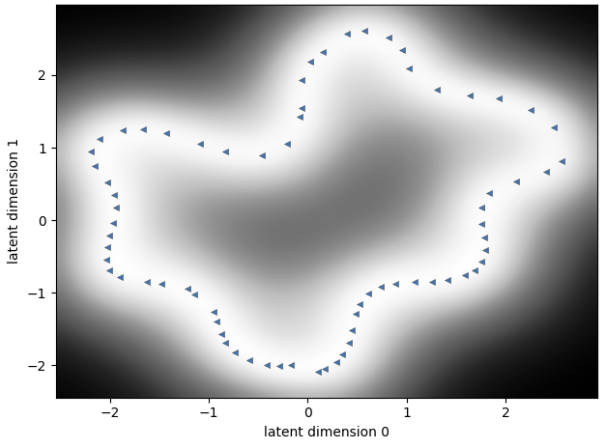


Figure 5.3: Two-dimensional latent representation learned by the GPLVM for the COIL Lucky Cat dataset. Each blue marker corresponds to a latent embedding of one image in the sequence. The background shading visualises the estimated uncertainty or density induced by the GPLVM mapping, revealing a smooth, closed nonlinear manifold structure in latent space.

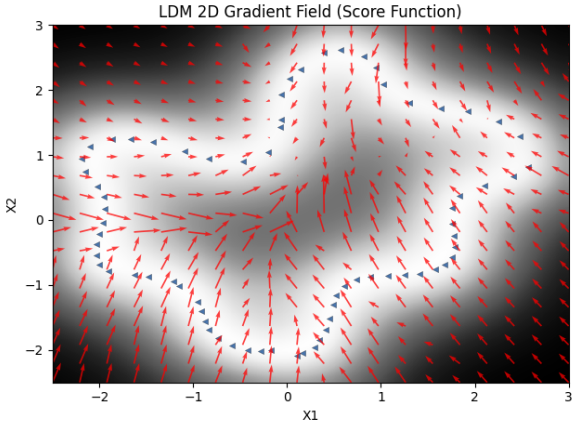


Figure 5.4: Coil dataset vector field generated by LDM

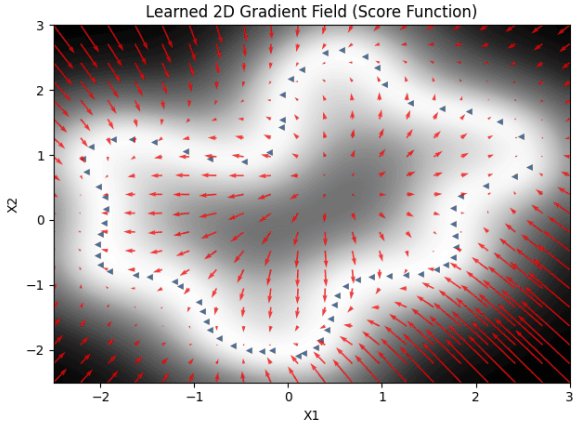


Figure 5.5: Coil dataset vector field generated by ILDM

To examine how these differences in the learned score fields translate into actual generative performance, we compare the samples produced by different diffusion models. Figure 5.6 shows samples generated by a score based diffusion model [41], which largely fail to produce meaningful structure and resemble noise. Figure 5.7 presents samples generated by a standard LDM that performs diffusion in a Euclidean latent space without geometric awareness. These samples exhibit visible distortions and artifacts, particularly in finer details.

Figure 5.8 presents samples generated by ILDM with the approximate DSM objective. The σ^2_{thresh} of this samples is 0.04. The reconstructions preserve the global structure, shape, and color consistency of the original object across multiple views, indicating that ILDM successfully captures the data distribution on the unknown manifold.

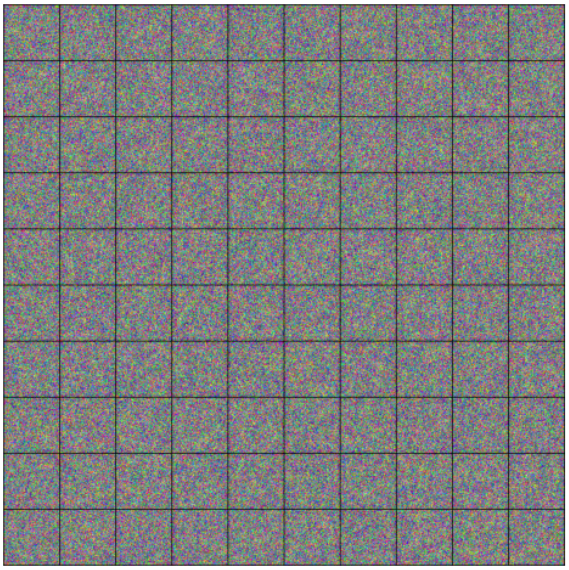


Figure 5.6: 100 coil images graphs generated by Diffusion Model

Figure 5.7: 100 coil images graphs generated by LDM

In section 4.2, we mentioned that a scaling parameter α is needed in forward and backward process of ILDM, where $\sigma^2_{\text{thresh}} = \alpha\,\sigma^2_{\text{max}}$. To evaluate the effect of the scaling parameter α on the performance of the ILDM framework, we conducted experiments using different benchmark datasets. Specifically, we tested three threshold settings corresponding to 30%, 40%, and 50% of the maximum variance of the mapping σ^2_{max}. For each setting, we generated samples using the proposed ILDM method and computed the FID to quantify generation quality. These results were compared against the FID obtained from a baseline Latent Diffusion Model (LDM). Table 5.2 summarises how different choices of the scaling parameter α affect the FID performance of ILDM.

Table 5.2: Effect of scaling parameter α on ILDM performance.(averaged over 10 independent runs)

α (%)	50	40	30	20	10	LDM
MEAN±sd FID↓	159±1.5	154.1±1.45	155±1.38	152±2.23	145±2.77	163±2.5

The FID values are compared against a baseline Latent Diffusion Model (LDM), showing that ILDM achieves lower FID when α is moderately small (e.g., $\alpha = 0.4$). The corresponding distribution of sampling points in the latent space is shown in Figure 5.9 and Figure 5.10. In both figures, the red points represent the ground truth data distribution, while the blue points denote the samples produced by the corresponding models. The sampling points generated by ILDM are observed to be closely distributed around the latent points, whereas those produced by LDM exhibit a comparatively larger deviation from the latent points, which indicates that ILDM samples are closer to the original graph.

Figure 5.8: 100 graphs generated by ILDM

Table 5.3: Performance comparison of diffusion models on the COIL image dataset. Evaluation metrics include FID and LPIPS, where lower values indicate better image quality and perceptual similarity.

Method	**ILDM**	**LDM**	**DM**
MEAN±SD FID↓	143±2.77	163±2.5	352.17±6.5
MEAN±SD LPIPS↓	0.5049±0.02	0.5442±0.03	1.2905±0.09

We further assess model performance using quantitative measures. Table 5.3 summarises results for three models: ILDM, standard LDM, and a baseline image-space diffusion model (DM). All models generate 200 samples 10 times, which are evaluated using FID and LPIPS. Lower FID and LPIPS scores indicate better visual fidelity and perceptual similarity, respectively. Among the three models, the ILDM achieves the best overall performance, obtaining the lowest FID (154.12) and LPIPS (0.5049) scores. The LDM performs moderately well but suffers from degraded perceptual quality. The baseline DM yields significantly higher FID and LPIPS values, indicating less effective image synthesis under the same evaluation. Since the generated graphs consist primarily of noise, we do not report the standard deviation.

5.2.1 VAE Experience

In the previous section, we constructed a latent representation of the COIL image dataset using a GPLVM. In this section, we instead employ a VAE as an alternative dimensionality

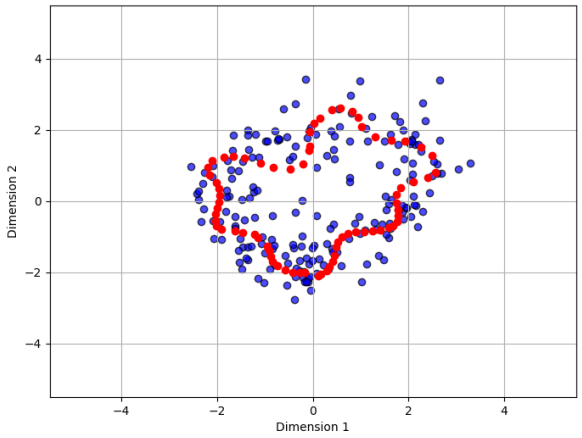


Figure 5.9: Samples in the latent space. Red points represent the original latent representations, while blue points denote samples generated directly in the latent space by the LDM. The comparison illustrates how well the generated samples align with the underlying latent structure.

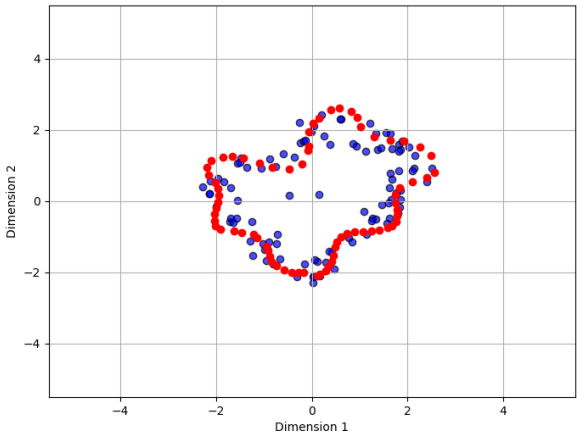


Figure 5.10: Samples in the latent space. Red points represent the original latent representations, while blue points denote samples generated directly in the latent space by the ILDM. The comparison illustrates how well the generated samples align with the underlying latent structure.

reduction method, and re-learn the latent space to examine how our generative models behave under a different latent parameterisation. The theoretical foundations of the Variational Autoencoder (VAE) have been presented in Section 2.3. An example of the VAE latent space learned from the COIL dataset is shown in Figure 5.11.

As introduced in Section 4.2, our forward process comprises two diffusion mechanisms. Although the uncertainty associated with the VAE mapping is not explicitly modelled, we introduce a switch between Euclidean Brownian motion (BM) and manifold-based BM, determined by the local data density in the latent space. To estimate this density, we apply Gaussian Kernel Density Estimation (KDE), a non-parametric technique that constructs a smooth estimate of the probability density function by averaging Gaussian kernels centered at each observation [83]. This allows us to define high-confidence regions in the latent space by thresholding the estimated density. Specifically, we define a high-confidence region by thresholding the KDE-estimated density at the 10th percentile. It help us to distinguish regions where the manifold structure is most reliable. Figure 5.12 visualises the KDE-estimated density over the VAE latent space, highlighting high-confidence regions where the learned manifold geometry is well supported by data and low-density regions where Euclidean diffusion is preferred.

Figure 5.13 compares the reconstruction quality of COIL Lucky Cat images obtained using ILDM and LDM within the VAE latent space. The samples reconstructed by VAE ILDM (Fig. 5.13a) exhibit clearer object contours, more consistent color patterns, and smoother viewpoint transitions across the grid. In contrast, the reconstructions produced

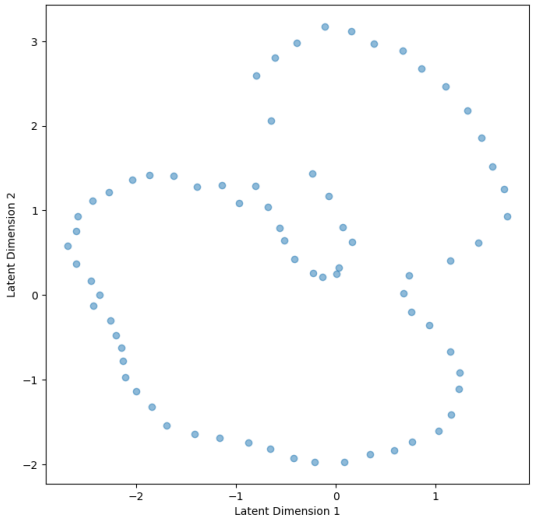


Figure 5.11: Two-dimensional latent representation learned by the VAE for the COIL Lucky Cat dataset. Each blue point corresponds to the latent embedding of one image.

by the VAE LDM (Fig. 5.13b) show noticeable artifacts, including local distortions and irregular texture patterns, particularly around fine structural details such as facial features and surface decorations. These visual differences indicate that, even under the same VAE latent parameterisation, incorporating intrinsic geometric structure into the diffusion dynamics leads to more stable and faithful reconstructions.

(a) VAE ILDM generated images

(b) VAE LDM generated images

Figure 5.13: Reconstruction results of COIL images using different methods under VAE latent space.

Table 5.4 presents a quantitative comparison of three generative methods ILDM, LDM, and DM on the COIL dataset, evaluated within the VAE latent space 10 times.

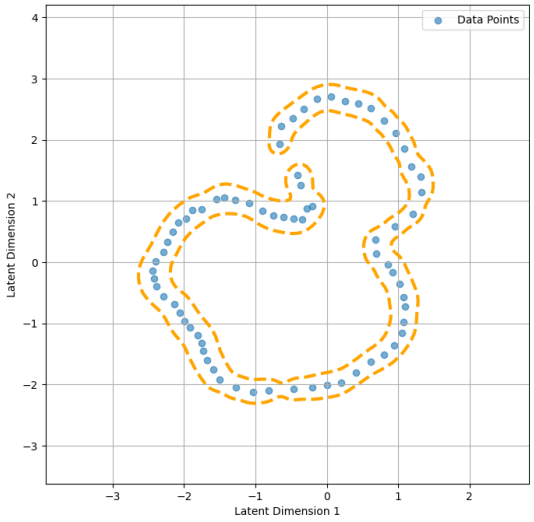


Figure 5.12: Kernel density estimate (KDE) of the VAE latent space for the COIL Lucky Cat dataset. Blue points denote latent representations of the data, and the orange curve indicates the 10% density level set.

FID and LPIPS are used to measure the fidelity and perceptual similarity of generated images to real ones. Among the three methods, ILDM consistently outperforms the others, achieving the lowest scores across all metrics. Examples of generated images from ILDM and LDM are shown in the Figure.

Table 5.4: Quantitative comparison of generative methods (ILDM, LDM, and DM) in the VAE latent space using FID and LPIPS on the COIL dataset. Lower values indicate better performance.

Method	**ILDM**	**LDM**
MEAN±SD FID↓	207±3.2	283.11±3.9
MEAN±SD LPIPS	0.4966±0.012	0.6359±0.02

5.2.2 Condition

The coil dataset was divided into two groups based on orientation: face-forward and face-backward, as illustrated in Figure 5.14. A class-conditioning mechanism was incorporated into the ScoreNet by employing a conditional multi-layer perceptron (MLP) and conditional generation is implemented using classifier-free guidance (Section 4.5).

The figure 5.15 presents a two-dimensional latent space representation learned by the model under two different conditions. Each point corresponds to a latent embedding, with blue points representing the face-forward condition and red points representing the face-backward condition of the Lucky Cat object. The clear separation between the two clusters indicates that the model successfully distinguishes between the two orientations,

Figure 5.14: Face-forward and Face-backward Lucky Cat examples

capturing meaningful pose-dependent variations in the latent space.

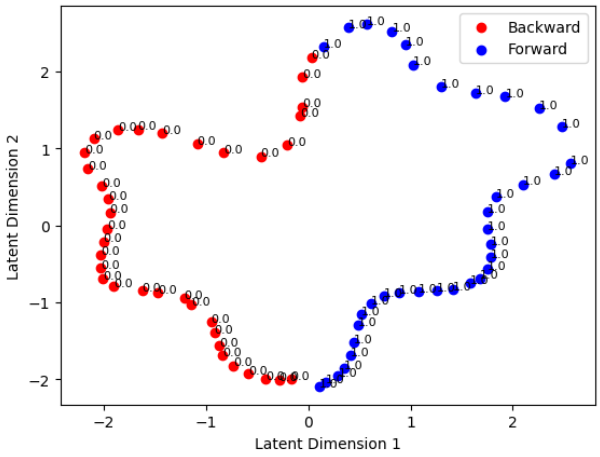


Figure 5.15: Latent space visualisation of the Lucky Cat under two conditions: face-forward (blue) and face-backward (red). The model effectively separates the two orientations, indicating distinct latent representations for each pose.

During training, the conditioning input c is randomly set to null with probability p_{uncond}, allowing the model to simultaneously learn both conditional and unconditional score functions. As demonstrated in [40, 45], small values of $p_{\text{uncond}} \in \{0.1, 0.2\}$ are sufficient for effective guidance without compromising conditional model performance. We adopt a similar setting and find it adequate for high-quality guided sampling within our ILDM framework. Detailed algorithms for training the joint score network are provided in Section 4.5. We also give a score network structure in Table 5.5. A comparison of the conditional vector fields produced by LDM and ILDM is presented in Figure 5.16 and Figure 5.17.

Table 5.5 outlines the structure of the proposed training model, which integrates angle and time-step embeddings into a U-Net backbone for condition-aware feature extraction and reconstruction.

Compared to the LDM results in Figure 5.16, the ILDM visualisation in Figure 5.17 exhibits smoother transitions and more coherent vector directions, suggesting that ILDM captures a more stable and well-structured latent representation. This demonstrates that

Table 5.5: Architecture of the proposed conditional training model, which enhances a U-Net with angle and time-step embeddings.

Module	Input Shape	Operation	Output Shape	Description
Input	(B, 1, 2)	–	(B, 1, 2)	Main input tensor x.
Angle Condition (cond)	(B, 2)	MLP: Linear(2→128) + ReLU + Linear(128→128) + ReLU	(B, 128)	Angle condition embedding.
Time Condition (t)	(B, 1)	Linear(1→128) + ReLU + Linear(128→128) + ReLU	(B, 128)	Time-step embedding.
Concatenation	(B, 1, 2) + (B, 128, 2, 2) + (B, 128, 2, 2)	Channel concatenation	(B, 257, 2, 2)	Combine input and conditions.
Encoder Block 1	(B, 257, 2, 2)	Conv2D(257→64, k=3, p=1) + ReLU	(B, 64, 2, 2)	First convolutional encoder layer.
Encoder Block 2	(B, 64, 2, 2)	Conv2D(64→128, k=3, s=1, p=1) + ReLU	(B, 128, 2, 2)	Second convolutional encoder layer.
Bottleneck	(B, 128, 2, 2)	Conv2D(128→128, k=3, p=1) + ReLU	(B, 128, 2, 2)	Latent feature extraction.
Decoder Block 1	(B, 128, 2, 2)	ConvTranspose2D(128→64, k=4, s=2, p=1) + ReLU	(B, 64, 4, 4)	Upsampling through transposed convolution.
Decoder Block 2	(B, 64, 4, 4)	Conv2D(64→3, k=3, p=1) + Tanh	(B, 3, 4, 4)	Reconstruct output feature map.
Flatten + Select	(B, 3, 4, 4)	Flatten → Slice [:, :2]	(B, 2)	Take first two dimensions as final output.

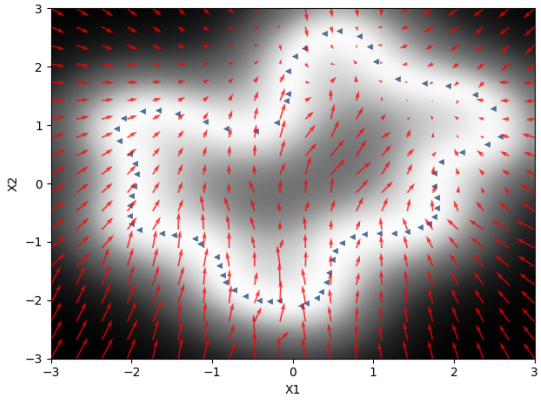


Figure 5.16: Gradient Field generated by LDM

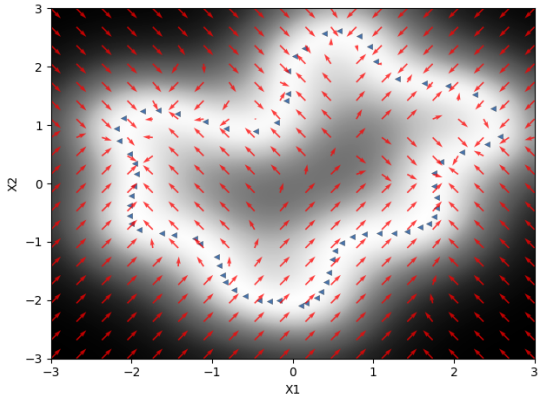


Figure 5.17: Gradient Field generated by ILDM

ILDM provides a more consistent modeling of the underlying data distribution than the standard LDM. As shown in Figure 5.18, the LDM-generated samples display noticeable dispersion, with several points deviating from the target distribution. In contrast, Figure 5.19 shows that ILDM produces samples more closely aligned with the ground truth, forming a tighter and more coherent pattern. This comparison indicates that ILDM generates higher-quality samples and better preserves the underlying data structure than LDM.

Figure 5.20 and Figure 5.21 present image samples generated by LDM and ILDM. Figure 5.20 show noticeable artifacts and incomplete structures, with several images appearing noisy or partially missing content. In contrast, the samples produced by ILDM in Figure 5.21 demonstrate greater visual consistency and clarity, with more complete object representations and fewer distortions. These results indicate that ILDM enhances the generation quality by improving stability and fidelity in the reconstructed images compared to the standard LDM.

Here we assess model performance using quantitative measures. Table 5.6 summarises results for ILDM and LDM. All models generate 200 samples 10 times, which are evaluated using FID and LIPIPS.

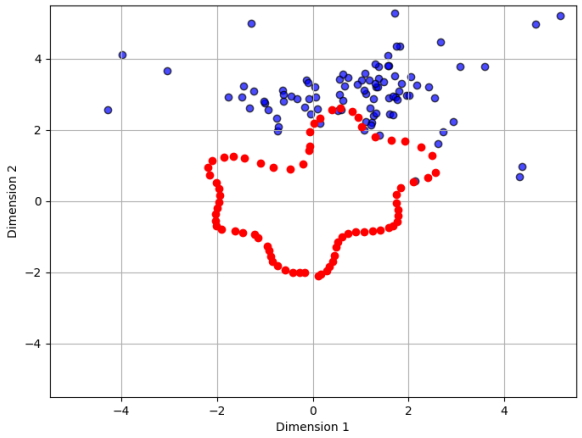


Figure 5.18: Samples generated by LDM

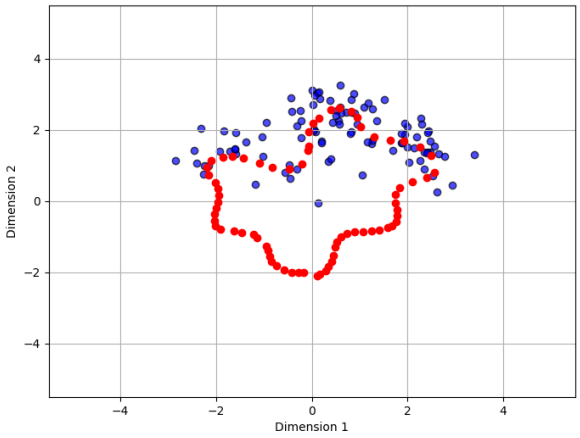


Figure 5.19: Samples generated by ILDM

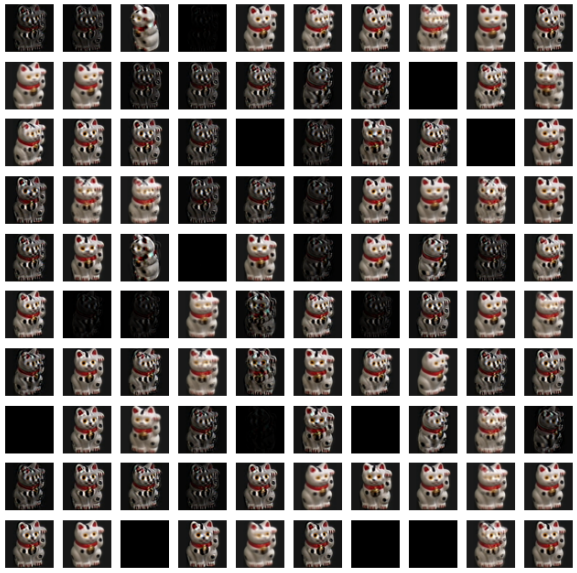


Figure 5.20: 100 images generated by LDM

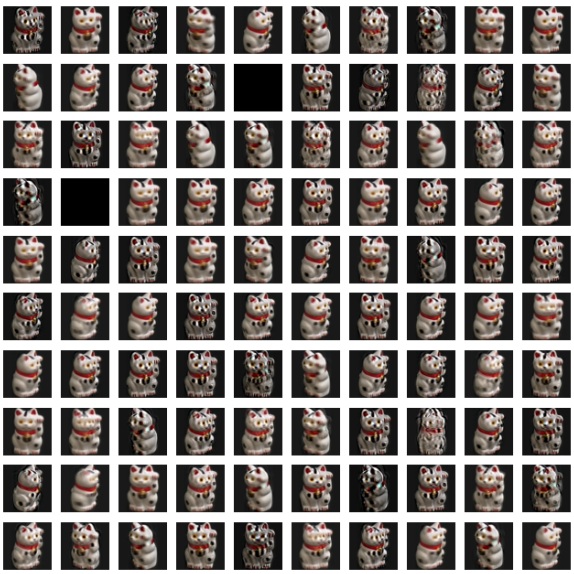


Figure 5.21: 100 images generated by ILDM

Table 5.6: Performance comparison of condition diffusion models on the COIL image dataset. Evaluation metrics include FID and LPIPS, where lower values indicate better image quality and perceptual similarity.

Method	**LDM**	**ILDM**
MEAN±SD FID↓	184±1.9	122±1.5
MEAN±SD LPIPS↓	0.5813±0.0062	0.4588±0.0089

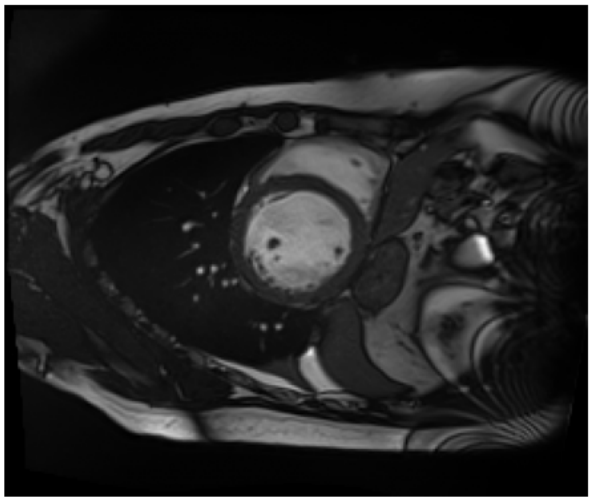
5.3 Left Ventricle Cardiac MRI further results

We then evaluate the proposed ILDM on a cardiac magnetic resonance image (MRI) dataset focusing on the left ventricle. It is used to show that the method also works on

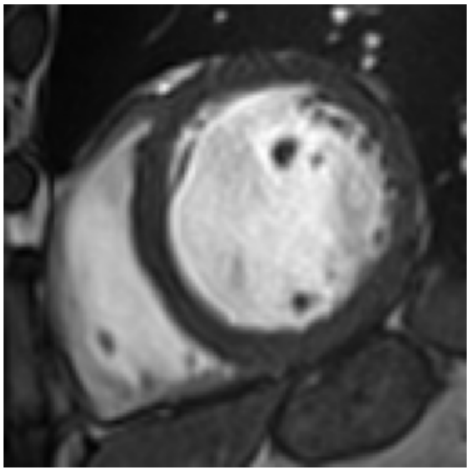
more realistic and complex medical data (cardiac MRI). The dataset is constructed from two publicly available benchmarks: the Automatic Cardiac Diagnosis Challenge (ACDC) and the MICCAI 2020 Left Ventricle Challenge [53, 54]. It comprises 55 healthy subjects and 55 subjects diagnosed with hypertrophic cardiomyopathy (HCM), a structural heart disease characterised by abnormal myocardial thickening. For each subject, three short-axis MRI slices are selected, resulting in a total of 330 images.

All images are resamplled to a uniform spatial resolution of 128×128 pixels and cropped around the left ventricular region to isolate the cardiac anatomy from surrounding organs and background tissues, as illustrated in Figure 5.22. This preprocessing step not only focuses the analysis on clinically relevant structural variations but also reduces extraneous variability, thereby facilitating the learning of a compact and geometrically meaningful latent representation. The slices are further aligned to ensure consistent orientation across subjects, enabling the latent space to more faithfully capture intrinsic anatomical variability.

The preprocessed images are subsequently encoded into a five-dimensional latent space for diffusion modeling. Owing to the constrained anatomical structure of the left ventricle and the limited degrees of morphological variation, the resulting data distribution exhibits a low-dimensional manifold structure embedded in the high-dimensional image space, making this dataset particularly suitable for evaluating geometry-aware generative modeling. 10 examples of the cropped dataset are shown in Figure 5.23.



(a) An example of original input image



(b) Corresponding cropped image

Figure 5.22: ACDC dataset pre-prosessing

To determine the effective intrinsic dimensionality of the cardiac MRI data, we further perform an Automatic Relevance Determination (ARD) analysis on the learned B-GPLVM latent space. Figure 5.24 shows the relevance contribution of each latent dimension. The

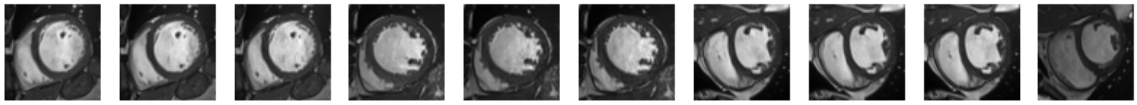


Figure 5.23: 10 examples of cropped heart dataset.

first five dimensions exhibit substantially higher relevance values compared to the remaining dimensions, while the contributions of higher dimensions rapidly decay toward negligible values. This indicates that the essential variability of the cardiac MRI data is well captured by a five-dimensional latent manifold, which justifies the choice of a five-dimensional intrinsic latent representation for subsequent diffusion modeling.

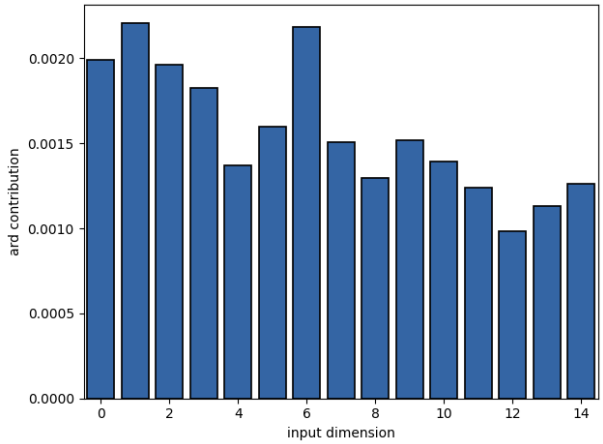


Figure 5.24: Automatic Relevance Determination (ARD) analysis of the learned GPLVM latent space for the heart dataset. The bar heights indicate the relevance contribution of each latent dimension.

Figure 5.25 illustrates the latent representation of the ACDC dataset obtained from the trained GPLVM model. The original latent space has five dimensions, but for visualisation purposes, only the first three dimensions are displayed.

The ILDM model was trained for 400 epochs using the Adam optimiser with a learning rate of 1×10^{-4}. For Brownian motion path sampling, we use 100 paths, each with 100 steps and a step size of 1, resulting in a total diffusion time of 100, and generate samples in a 5-dimensional latent space with a variance threshold of 0.02. During training, the ScoreNet learned to approximate the time-dependent score function by minimizing a diffusion-based reconstruction loss. After training, 100 latent samples were generated via an SDE-based sampling process, which integrates the learned score field backward from Gaussian noise. This sampling procedure ensures that each generated point follows the model's estimated latent dynamics, capturing both deterministic and stochastic components of the diffusion process.

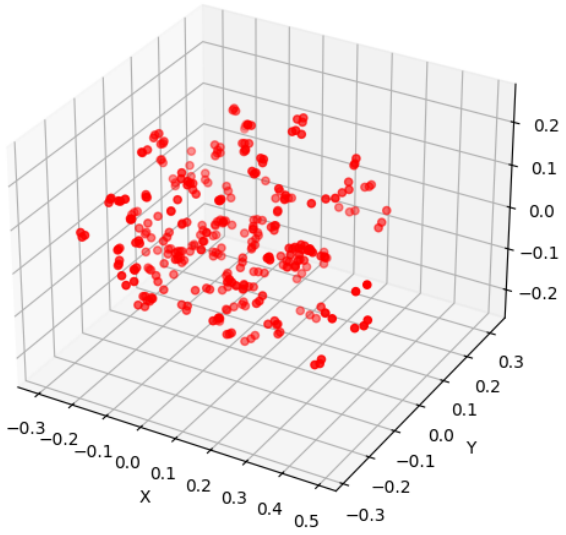


Figure 5.25: First three dimensions Visualisation of the latent space representation for the ACDC dataset. A total of 330 latent points are plotted, revealing the distribution and compactness of the learned representations.

Table 5.7 summarises the time-conditioned 1D U-Net used in ILDM for the ACDC dataset, where temporal embeddings are injected at each layer to guide feature extraction and reconstruction.

Table 5.7: Architecture of the 1D U-Net used in ILDM for the ACDC dataset.

Module	Input Shape	Operation	Output Shape	Description
Input	(B, 1, L)	–	(B, 1, L)	1D input signal.
Time Embedding	(B, 1)	Gaussian Fourier Projection + Linear(256) + SiLU	(B, 256)	Generates a time-dependent embedding vector.
Encoder Block 1	(B, 1, L)	Conv1D(1→32, k=3, p=1) + Dense(256→32) + GroupNorm(4) + SiLU	(B, 32, L)	Extracts local temporal features conditioned on t.
Encoder Block 2	(B, 32, L)	Conv1D(32→128, k=3, s=2, p=1) + Dense(256→128) + GroupNorm(32) + SiLU	(B, 128, L/2)	Downsamples feature maps while injecting time embedding.
Encoder Block 3	(B, 128, L/2)	Conv1D(128→256, k=3, s=2, p=1) + Dense(256→256) + GroupNorm(32) + SiLU	(B, 256, L/4)	Deep encoder representation.
Decoder Block 3	(B, 256, L/4)	ConvT1D(256→128, k=3, s=2, p=1, op=1) + Dense(256→128) + GroupNorm(32) + SiLU	(B, 128, L/2)	First upsampling stage with skip connection to Encoder2.
Decoder Block 2	(B, 128+128, L/2)	ConvT1D(256→32, k=3, s=2, p=1, op=1) + Dense(256→32) + GroupNorm(32) + SiLU	(B, 32, L)	Second upsampling stage with skip connection to Encoder1.
Decoder Block 1	(B, 32+32, L)	ConvT1D(64→1, k=3, p=1)	(B, 1, L)	Output reconstruction layer.
Normalisation	(B, 1, L)	Elementwise division by σ_t^2	(B, 1, L)	Normalises output by time-dependent noise variance.

To examine how the learned latent geometry influences the behavior of the diffusion dynamics and the resulting sample distributions, we next visualise the latent samples generated by LDM and ILDM. Figures 5.26 and 5.27 display three-dimensional visualisations of LDM and the ILDM. While the latent space is five-dimensional, these plots project the data onto three dimensions (Dimension 1, Dimension 2, and Dimension 3) for visualisation purposes. In both figures, the red points represent the first three dimension of true latent points, while the blue points denote the generated samples from the respective model. Figure 5.26 (LDM samples) shows a distribution that is relatively scattered, with points spread across a wide volume of the space. In contrast, Figure 5.27 (ILDM samples) displays a notably compact and centralised distribution. The points form a tight, spherical cluster around the true latent points. This comparison visually indicates that the ILDM generates a significantly more constrained and concentrated latent representation than the

LDM, suggesting a lower overall variance in its samplled parameters.

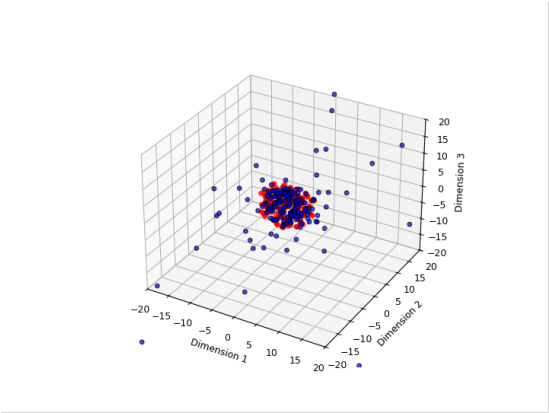


Figure 5.26: Three-dimensional projections of latent samples generated by the LDM. 200 samples exhibit noticeable dispersion in latent space.

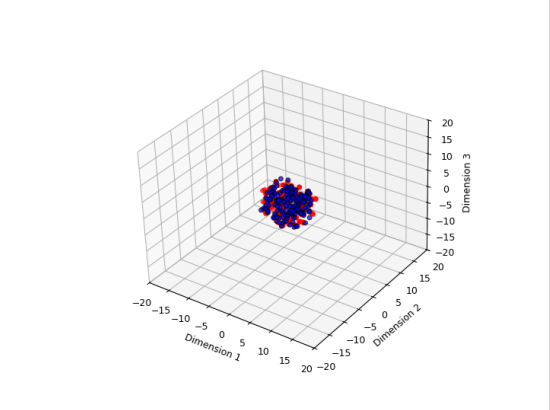


Figure 5.27: Three-dimensional projections of latent samples generated by the ILDM. 200 samples form a compact cluster closely aligned with the latent data distribution.

We next examine how these differences in latent sampling behavior translate into graph generative performance. Figures 5.28, 5.29, and 5.30 show the generative performance across three models: DM, LDM, and ILDM. Figure 5.29 highlights the challenge of pixel-space modeling, as the DM fails to produce coherent structure and instead yields unstructured artifacts. Figure 5.28, representing the standard LDM, successfully generates recognizable cardiac images; however, the output suffers from noticeable artifacts, inconsistencies (e.g., the clear generation error block), and a general lack of uniform crispness. In contrast, Figure 5.30 shows the results generated by the ILDM. These samples are noticeably higher in quality — they are clear, realistically shaped, and consistent across the entire grid. This highlights how ILDM's constrained latent space helps it produce reliable and high-quality images.

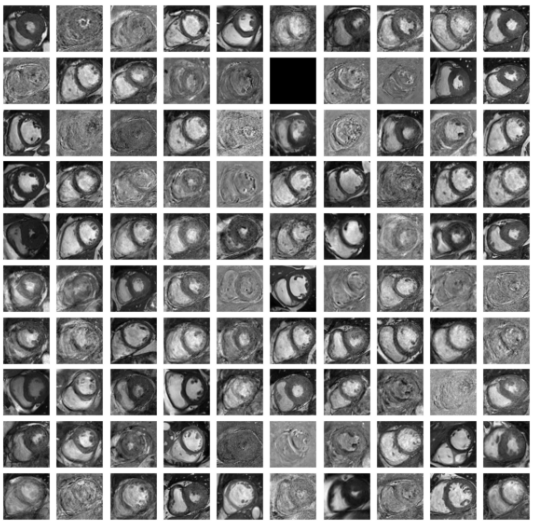


Figure 5.28: Images generated by LDM.

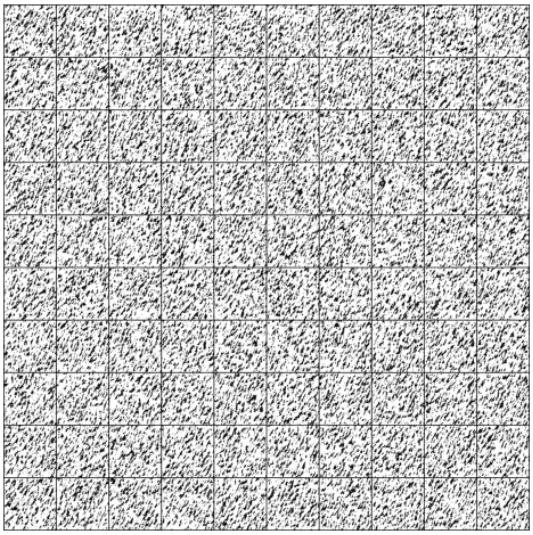


Figure 5.29: Images generated by DM.

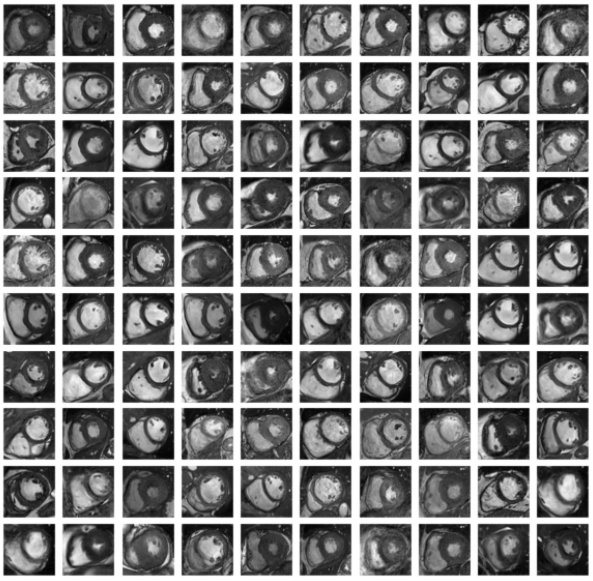


Figure 5.30: Images generated by ILDM

Table 5.8 presents the quantitative results. We generate 300 images for 10 times. ILDM achieves the best overall performance, yielding the lowest FID score (162.94) and LPIPS score (0.5118). These results highlight ILDM's ability to more accurately model the structural and visual characteristics of cardiac MRI data compared to LDM and DM.

We also give an ablation study of the scaling parameter α in Table 5.9. We tested three threshold settings corresponding to 20%, 25%, 30% and 40% of the maximum variance of the mapping σ^2_{max}. Due to computational limitations, each setting generates 1000 samples only once, and therefore no repeated experiments were performed to estimate

Table 5.8: Performance comparison on the left ventricle cardiac MRI dataset.

Method	**ILDM**	**LDM**	**DM**
MEAN±SD FID ↓	160±2.1	179±3.3	404.53±5.9
MEAN±SD LPIPS ↓	0.5118±0.031	0.5278±0.033	0.7241±0.046

the mean or standard deviation. The best FID score is achieved when α is set to 40%

Table 5.9: Effect of scaling parameter α on ILDM performance.

α (%)	50	40	35	30	LDM
MEAN FID↓	166	162	173	199	221

($\sigma^2_{\text{thresh}} = 0.03$), indicating that this level of latent space scaling provides the most effective balance for high-quality image synthesis. Compared to the baseline LDM, ILDM at this setting clearly produces superior results.

5.3.1 Condition

The cardiac MRI dataset was divided into two groups: healthy subjects and hypertrophic cardiomyopathy (HCM) subjects, as illustrated in Figure 5.31 and Figure 5.32.

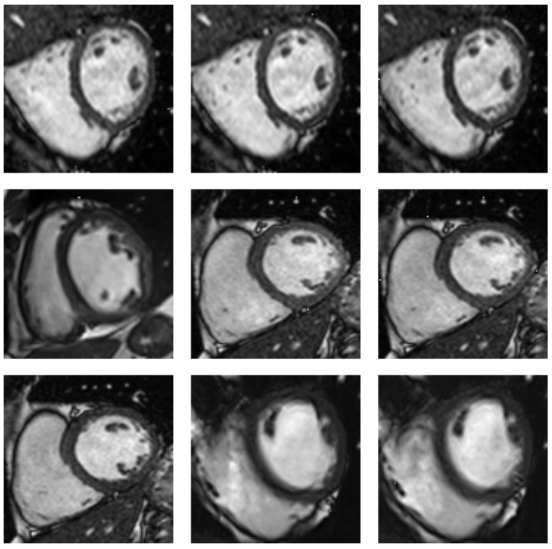


Figure 5.31: 9 examples of normal cardiac MRI images, displaying a symmetric ventricular structure and uniform myocardial wall thickness.

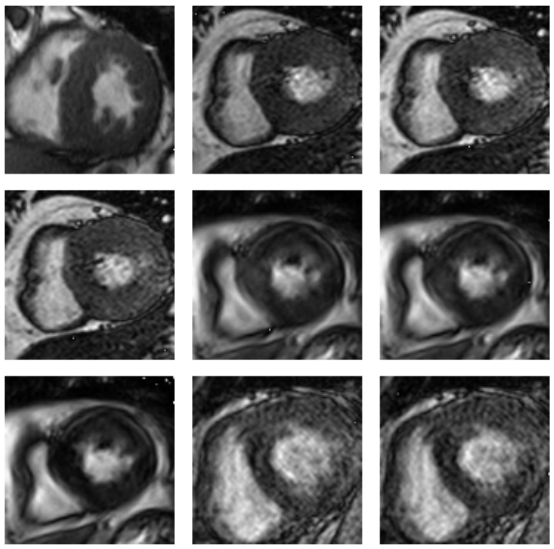


Figure 5.32: 9 examples of HCM cardiac MRI images, showing asymmetric myocardial hypertrophy, particularly with noticeable thickening of the interventricular septum.

The examples in Figure 5.31 show healthy myocardium characterised by a symmetrical and uniform thickness of the ventricular walls and a regular chamber shape. In

contrast, Figure 5.32 clearly demonstrates the pathological manifestation of HCM, specifically exhibiting asymmetric hypertrophy, where certain regions of the myocardial wall, most notably the interventricular septum, are significantly and disproportionately thickened. This difference highlights the critical structural variations the models must learn to accurately generate, from the balanced geometry of a normal heart to the distorted, thickened morphology of an HCM-affected heart.

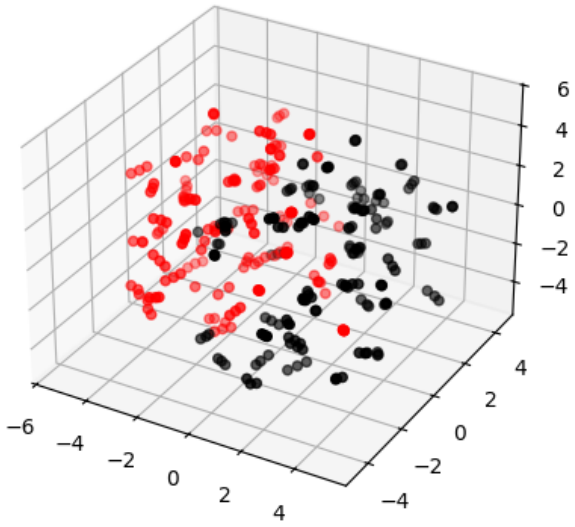


Figure 5.33: Latent Space Distribution of Cardiac Classes. A three-dimensional projection of the five-dimensional latent space showing the distribution of the two classes: red points represent the Normal heart condition, and black points represent the Hypertrophic Cardiomyopathy (HCM) condition.

Figure 5.33 displays the three-dimensional projection of the cardiac dataset's five-dimensional latent space, where red points represent Normal hearts and black points represent HCM hearts. The visualisation reveals that the two classes form distinct clusters which are largely separable. Specifically, the red (Normal) points primarily occupy the left and rear regions of the space, while the black (HCM) points form a separate volume toward the right and front. Although a minor overlap exists at the interface, the two distributions exhibit clear bimodal separation, indicating that the latent representation successfully encodes the structural differences between the Normal and HCM heart conditions. This separability suggests that the latent space provides a robust foundation for both classification and conditional generation tasks.

To incorporate categorical information and diffusion time into the reconstruction process, a class-conditioned U-Net architecture (UNetWithT) is employed. Table 5.10 summarises the conditional U-Net employed for the ACDC dataset, where class and time-step embeddings are injected via MLPs and broadcast before entering the encoder. The class

condition is represented as a binary array (e.g., $[0, 1]$ or $[1, 0]$) and embedded through the MLP before being integrated into the network.

Table 5.10: Architecture of the conditional U-Net used for the ACDC dataset, incorporating class and time-step embeddings to enable condition-aware feature extraction.

Module	Input Shape	Operation	Output Shape	Description
Input	(B, 1, 5, 1)	–	(B, 1, 5, 1)	Main 5-dimensional input vector reshaped for 2D conv.
Class Condition MLP	(B, 2)	Linear(2→128) + ReLU + Linear(128→128) + ReLU	(B, 128)	Embeds class/angle condition.
Time Condition MLP	(B, 1)	Linear(1→128) + ReLU + Linear(128→128) + ReLU	(B, 128)	Time-step embedding.
Cond Broadcast	(B,128), (B,128)	Broadcast to spatial shape	(B,128,5,1)	Match input spatial dimensions before concatenation.
Concatenation	1 + 128 + 128 channels	Channel concat	(B, 257, 5, 1)	Merge input, angle cond, and time cond.
Encoder Block 1	(B,257,5,1)	Conv2D(257→64, k=3, p=1) + ReLU	(B,64,5,1)	First encoding layer.
Encoder Block 2	(B,64,5,1)	Conv2D(64→128, k=3, s=2, p=1) + ReLU	(B,128,3,1)	Downsampling stage.
Bottleneck	(B,128,3,1)	Conv2D(128→128, k=3, p=1) + ReLU	(B,128,3,1)	Latent feature extraction.
Decoder Block 1	(B,128,3,1)	ConvT2D(128→64, k=3, s=2, p=1) + ReLU	(B,64,5,1)	Upsampling layer.
Decoder Block 2	(B,64,5,1)	Conv2D(64→3, k=3, p=1) + Flatten + Tanh	(B,3,5,1) + flatten	Output activation before flattening.
Flatten + Select	(B,3×5)	Reshape + Slice [:, :5]	(B,5)	Select first five dimensions as final output.

Figures 5.34 and 5.35 show the sample distributions produced by the LDM and ILDM models, respectively, each using 200 sampling steps. In both plots, the red points represent the true latent representations of the underlying data, while the blue points correspond to the latent samples generated by each model. As shown in Figure 5.34, the samples produced by the LDM display noticeable dispersion around the true latent cluster, indicating a less precise approximation of the latent manifold. In contrast, Figure 5.35 demonstrates that the ILDM is able to generate samples that more closely align with the true latent points, forming a tighter cluster. This comparison highlights the improved capability of ILDM in capturing the intrinsic structure of the latent space.

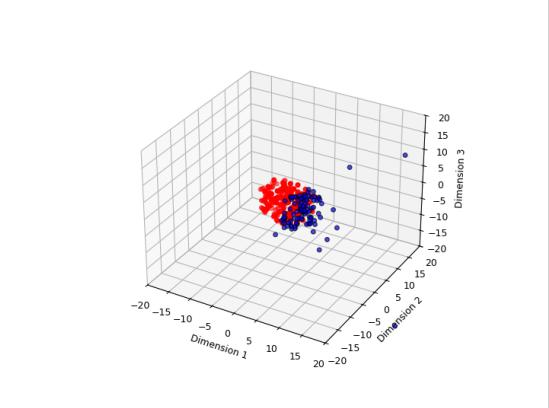


Figure 5.34: Latent space samples generated using LDM. Red points denote the true latent representations, while blue points indicate the samplled latent points.

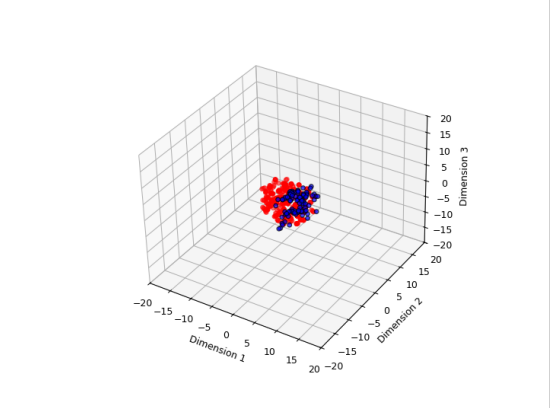


Figure 5.35: Latent space samples generated using ILDM. Red points denote the true latent representations, while blue points indicate the samplled latent points.

Figures 5.36 and 5.37 compare image samples generated by the LDM and ILDM models, respectively. In Figure 5.36, the images produced by the LDM exhibit noticeable variability in contrast and anatomical consistency, with several samples showing distortions

or incomplete structural formation. In contrast, the ILDM-generated samples in Figure 5.37 appear more stable and coherent, better preserving the characteristic shape and texture patterns of the target data distribution. This comparison indicates that ILDM achieves improved sample quality and structural fidelity relative to LDM.

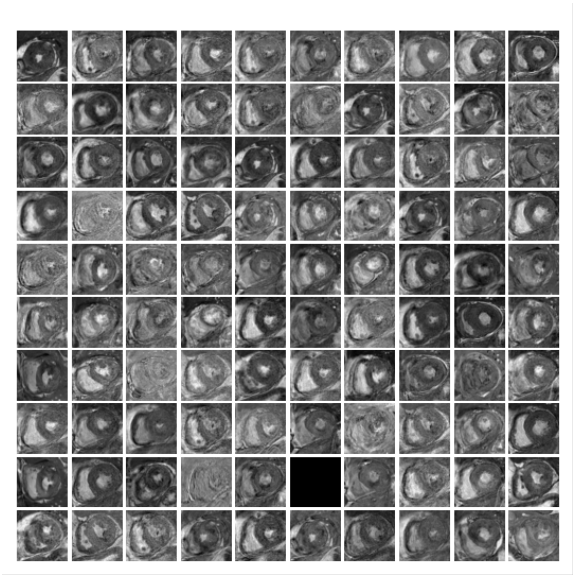


Figure 5.36: Random image samples generated using LDM. The generated images show varying degrees of structural fidelity and contrast consistency.

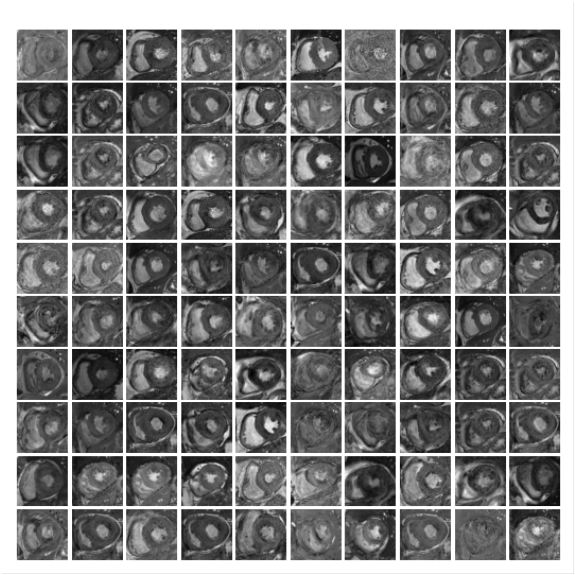


Figure 5.37: Random image samples generated using ILDM. The samples exhibit improved anatomical coherence and more consistent visual quality compared to LDM.

Metric	**ILDM**		**LDM**	
	nor	**hcm**	**nor**	**hcm**
MEAN±SD FID↓	205±2.1	230±2.2	219.81±2.4	242.37±2.8
MEAN±SD LPIPS ↓	0.5006±0.008	0.5181±0.011	0.5012±0.010	0.5189±0.011

Table 5.11: HEART: FID comparison with random starting noise (300 samples 10 times). LDM:10000 epochs; ILDM: 500 epochs; “nor” and “hcm” indicate respective categories.

Table 4.6 provides a quantitative evaluation of the ILDM against the standard LDM using FID and LPIPS metrics across "NOR" and "HCM" categories. The ILDM consistently achieves significantly lower Fréchet Inception Distance (FID) scores (205.04 and 228.13), indicating superior generated image quality compared to the standard LDM (219.81 and 242.37). Both models exhibit comparable performance on the Learned Perceptual Image Patch Similarity (LPIPS) metric, with scores around 0.50 to 0.52. This analysis confirms the advantage of the Riemannian LDM in generating more realistic images (lower FID).

5.4 MNIST

We further evaluate our method on a subset of the MNIST dataset [52], a widely used benchmark composed of grayscale images of handwritten digits (0–9), each with a resolution of 28×28 pixels, to further validate the effectiveness of our method on standard image generation tasks. For training purposes, we specifically extracted 120 images corresponding to digits 3, 8, and 9. Following data-driven estimates of the intrinsic dimensionality of MNIST reported in [57], the input images are embedded into a ten-dimensional latent space to capture the underlying manifold structure. Representative examples of the selected input images are illustrated in Figure 5.38.

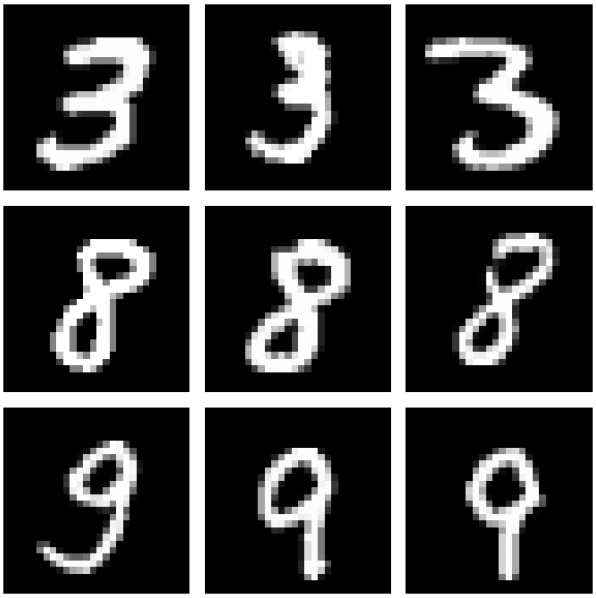


Figure 5.38: Examples of the training images

A Gaussian process (GP) decoder is pretrained to map the MNIST images into a ten-dimensional latent space, which serves as the intrinsic coordinate system for subsequent diffusion modeling. Based on this learned latent representation, we construct a geometry-aware diffusion model in latent space. Table 5.12 summarises the time-conditioned 1D U-Net employed in ILDM for the MNIST dataset, where Gaussian Fourier time embeddings are injected throughout the encoder–decoder pipeline.

Table 5.12: Architecture of the time-conditioned 1D U-Net used in ILDM for the MNIST dataset.

Module	Input Shape	Operation	Output Shape	Description
Input	(B, 1, L)	–	(B, 1, L)	1D input signal (e.g., latent trajectory).
Time Embedding	(B, 1)	Gaussian Fourier Projection + Linear(256) + Swish ($x \cdot \sigma(x)$)	(B, 256)	Time-step embedding vector.
Encoder Block 1	(B, 1, L)	Conv1D(1→32, k=3, s=1, p=1) + Dense(256→32) + GroupNorm(4) + Swish	(B, 32, L)	Base-resolution feature extraction with time conditioning.
Encoder Block 2	(B, 32, L)	Conv1D(32→128, k=3, s=2, p=1) + Dense(256→128) + GroupNorm(32) + Swish	(B, 128, L/2)	Downsampling and feature refinement.
Encoder Block 3	(B, 128, L/2)	Conv1D(128→256, k=3, s=2, p=1) + Dense(256→256) + GroupNorm(32) + Swish	(B, 256, L/4)	Deep latent representation at the bottleneck.
Decoder Block 3	(B, 256, L/4)	ConvT1D(256→128, k=3, s=2, p=1, op=1) + Dense(256→128) + GroupNorm(32) + Swish	(B, 128, L/2)	First upsampling stage with time-conditioned normalisation.
Decoder Block 2	(B, 128 + 128, L/2)	ConvT1D(256→32, k=3, s=2, p=1, op=1) + Dense(256→32) + GroupNorm(32) + Swish	(B, 32, L)	Second upsampling stage with skip connection from Encoder Block 1.
Decoder Block 1	(B, 32 + 32, L)	ConvT1D(64→1, k=3, s=1, p=1)	(B, 1, L)	Final reconstruction of the score field.
Normalisation	(B, 1, L)	Elementwise division by σ_t^2 via `get_std2(t)`	(B, 1, L)	Time-dependent variance normalisation of the output.

After training the score model, we analyse the resulting latent representations. During training, we use 100 Brownian motion paths with 100 steps and a step size of 0.5, resulting in a total diffusion time of 50. The model is trained on 120 samples in a 5-dimensional latent space with a variance threshold of 0.2. In the same way, the figure presents three-dimensional visualisations of LDM and the ILDM. While the latent space is ten-dimensional, these plots project the data onto three dimensions (Dimension 1, Dimension 2, and Dimension 3) for visualisation purposes. Figure 5.48 (LDM latent sample visualisation) and Figure 5.49 (ILDM latent sample visualisation) clearly demonstrate a difference in how each model distributes its samples in the projected space. The LDM samples are relatively dispersed across a larger volume, indicating a greater variance. In contrast, the ILDM samples (Figure 5.49) are tightly clustered and concentrated near the origin in a much smaller volume. This suggests the ILDM learns a more compact and efficient latent representation with lower variance.

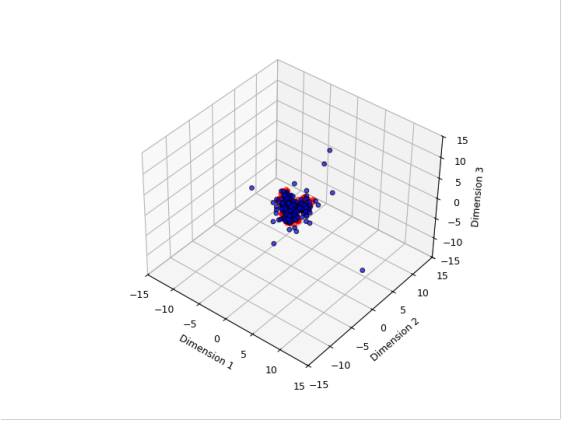


Figure 5.39: Three-dimensional projection of latent samples generated by the LDM. The samples are widely dispersed across the projected latent space, indicating high variance and weak concentration around the latent data manifold.

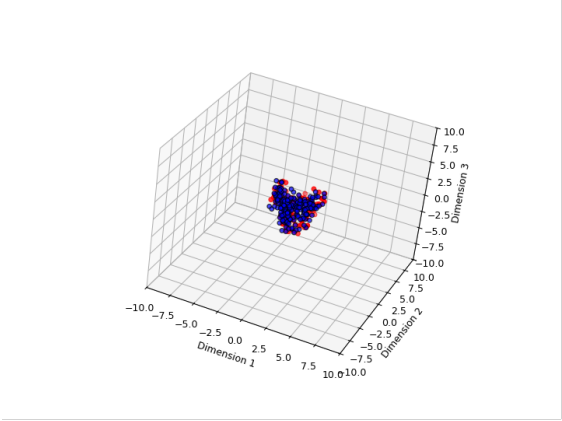


Figure 5.40: Three-dimensional projection of latent samples generated by the ILDM. The samples form a compact and well-centered cluster, reflecting a more structured and low-variance latent representation.

Figures 5.41 and 5.42 compare 100 graphs generated by LDM and ILDM. As shown in Figure 5.41, the samples from LDM are generally recognizable but often suffer from blurriness and structural distortions, indicating weaker latent representation quality. In contrast, Figure 5.42 demonstrates that ILDM produces clearer and more consistent digit shapes with fewer artifacts, suggesting that it learns a more robust and well-structured latent manifold for generation.

Figure 5.43 shows samples generated by the diffusion model (DM). The results appear highly noisy and lack coherent structure, indicating that the DM struggles to capture the underlying data distribution in this setting.

Figure 5.41: Samples generated by LDM. While the overall digit shapes are generally identifiable, many samples exhibit blurriness and structural artifacts, suggesting limitations in the latent representation quality.

Figure 5.42: Samples generated by ILDM. Compared with LDM, the generated digits are clearer and more consistent, indicating a stronger and more structured latent manifold learned by the model.

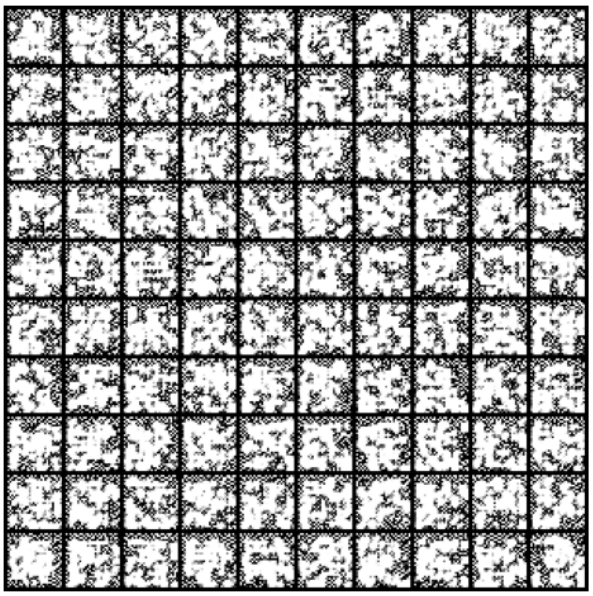


Figure 5.43: Samples generated by DM. The results are largely noisy and exhibit limited structural coherence, indicating poor generative performance in this setting.

Table 5.13 provides a quantitative evaluation using FID and LPIPS metrics. We generate 200 images for 10 times and take the average. The Riemannian LDM achieves the lowest FID score (136.70), outperforming both the standard LDM (155.93) and the baseline image-space DM (353.65). In terms of perceptual similarity, RLDM also performs

best, obtaining a slightly lower LPIPS value (0.6816) compared to LDM (0.6891). These results suggest that ILDM better captures the underlying digit structure and perceptual characteristics even in limited-data settings. We also give an ablation study of the scaling

Table 5.13: Performance comparison on a subset of the MNIST dataset.

Method	**ILDM**	**LDM**	**DM**
MEAN±SD FID ↓	135±2.4	166±2.8	353.65±4.9
MEAN±SD LPIPS ↓	0.6816±0.028	0.6891±0.032	0.8892±0.046

parameter α in Table 5.14. We tested three threshold settings corresponding to 15%, 20%, 30% and 40% of the maximum variance of the mapping σ^2_{max}:

Table 5.14: MNIST: Effect of scaling parameter α on ILDM performance.

α (%)	40	30	20	15	LDM
MEAN FID ↓	139	136	135	135	156

Due to computational constraints, each configuration was evaluated using a single run with 1000 generated samples, and no repeated experiments were conducted to compute mean and standard deviation statistics.

5.4.1 Condition

In this subsection, we divide the dataset into three distinct classes corresponding to the digits 3, 8, and 9(shown in below). Each image in the dataset is assigned to one of these categories based on its ground-truth label. This three-class configuration allows us to train and evaluate the model in a controlled classification setting, where the objective is to correctly identify which of the three digit classes each sample belongs to.

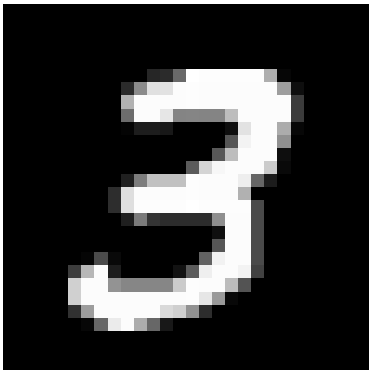


Figure 5.44: digit 3

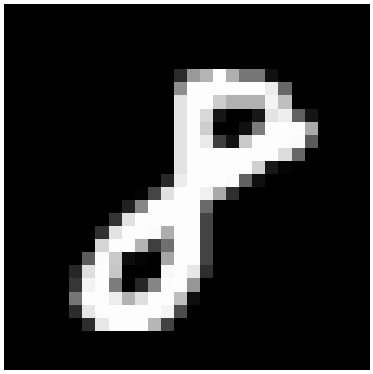


Figure 5.45: digit 8

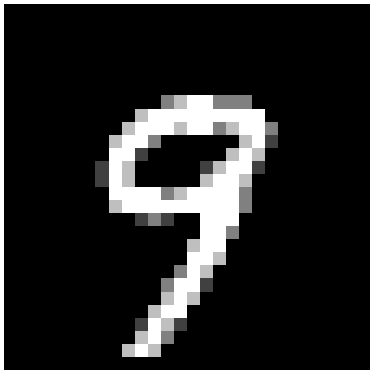


Figure 5.46: digit 9

Figure 5.47 contains 120 samples in total, with 40 instances each from digits 3 (blue),

8 (red), and 9 (black). Distinct clusters form in the projected space, illustrating class-dependent structure in the learned latent representation.

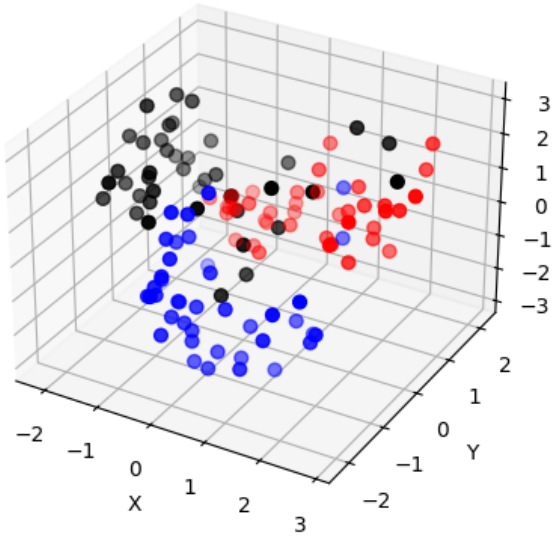


Figure 5.47: Visualisation of the first three dimensions of the 10-dimensional latent space.

We further extend the ScoreNet architecture by incorporating class-conditioning, where class labels and time embeddings are fused and injected into all layers of the network. Table 5.15 summarises the class-conditioned 1D U-Net used in our conditional diffusion experiments, where class and time-step embeddings are fused and injected into each layer of the encoder–decoder.

Table 5.15: Architecture of the class-conditioned time-aware 1D U-Net, where class and diffusion-time embeddings are fused and injected into all layers of the network.

Module	Input Shape	Operation	Output Shape	Description
Input	(B, 1, L)	–	(B, 1, L)	1D input signal.
Class Condition MLP	(B, 2)	Linear(2→256) + ReLU + Linear(256→256) + ReLU	(B, 256)	Embeds the class condition into the latent space.
Time Embedding	(B, 1)	Gaussian Fourier Projection + Linear(256)	(B, 256)	Encodes diffusion time step.
Fusion of Condition and Time	(B, 256), (B, 256)	Elementwise sum + Swish ($x \cdot \sigma(x)$)	(B, 256)	Produces joint time–class embedding used in all layers.
Encoder Block 1	(B, 1, L)	Conv1D(1→32, k=3, s=1, p=1) + Dense(256→32) + GroupNorm(4) + Swish	(B, 32, L)	Base-resolution feature extraction with conditioning.
Encoder Block 2	(B, 32, L)	Conv1D(32→128, k=3, s=2, p=1) + Dense(256→128) + GroupNorm(32) + Swish	(B, 128, L/2)	Downsampling and feature refinement.
Encoder Block 3	(B, 128, L/2)	Conv1D(128→256, k=3, s=2, p=1) + Dense(256→256) + GroupNorm(32) + Swish	(B, 256, L/4)	Deep latent representation at the bottleneck.
Decoder Block 3	(B, 256, L/4)	ConvT1D(256→128, k=3, s=2, p=1, op=1) + Dense(256→128) + GroupNorm(32) + Swish	(B, 128, L/2)	First upsampling stage with time–class conditioned normalisation.
Decoder Block 2	(B, 128+128, L/2)	ConvT1D(256→32, k=3, s=2, p=1, op=1) + Dense(256→32) + GroupNorm(32) + Swish	(B, 32, L)	Second upsampling stage with skip connection from Encoder Block 1.
Decoder Block 1	(B, 32+32, L)	ConvT1D(64→1, k=3, s=1, p=1)	(B, 1, L)	Final reconstruction of the conditional score field.
Normalisation	(B, 1, L)	Elementwise division by σ_t^2 via `get_std2(t)`	(B, 1, L)	Time-dependent variance normalisation of the output.

Figures 5.48 and 5.49 present conditional samples generated by the LDM and the proposed ILDM, respectively. Both visualisations show the first three dimensions of the latent space. The red points denote the latent points obtained from the GPLVM, while the blue points represent the points generated by sampling from the latent distribution.

In Figure 5.48, the samplled points (blue) generated by LDM are relatively more scattered around the latent points (red), indicating a less compact latent representation. In contrast, Figure 5.49 shows that ILDM produces samplled points that are more tightly

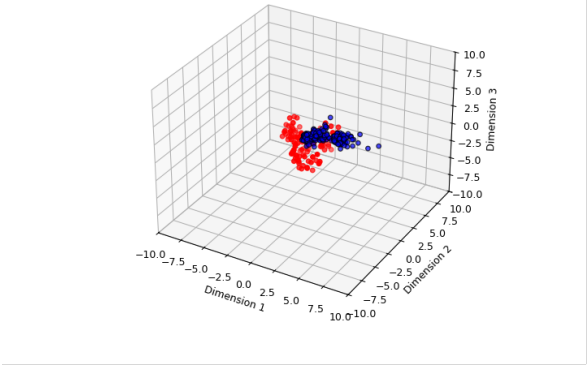


Figure 5.48: Conditioned latent samples generated by the proposed LDM.

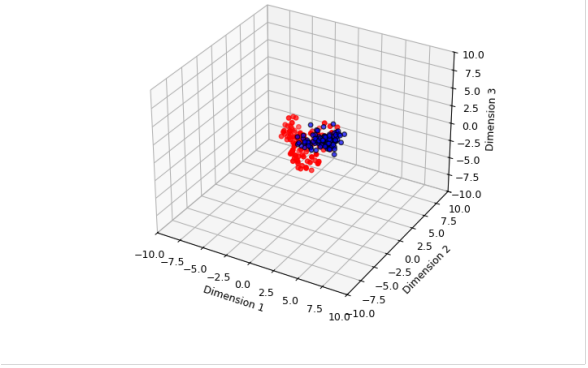


Figure 5.49: Conditioned latent samples generated by the proposed ILDM.

clustered around the latent points, suggesting that ILDM learns a more structured and stable latent space, leading to more consistent sampling performance.

Figures 5.50–5.53 show the generated MNIST digit samples using LDM and ILDM under different class conditions. Figures 5.50 and 5.51 correspond to digit "3," while Figures 5.52 and 5.53 correspond to digit "8." For both digit classes, the samples produced by ILDM exhibit more consistent digit shapes and fewer distorted or noisy patterns compared to LDM, demonstrating that ILDM is able to generate visually cleaner and more stable samples while preserving class-specific characteristics.

Figure 5.50: Samples generated by LDM.

Figure 5.51: Samples generated by ILDM.

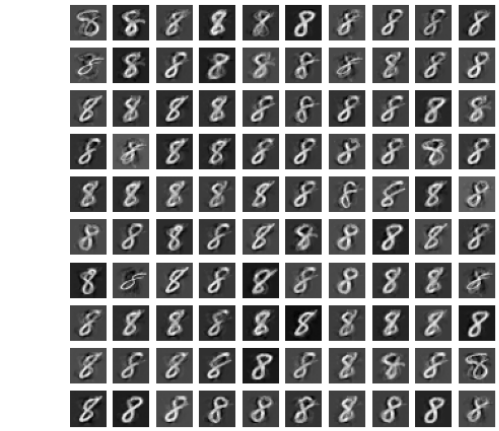


Figure 5.52: Samples generated by LDM.

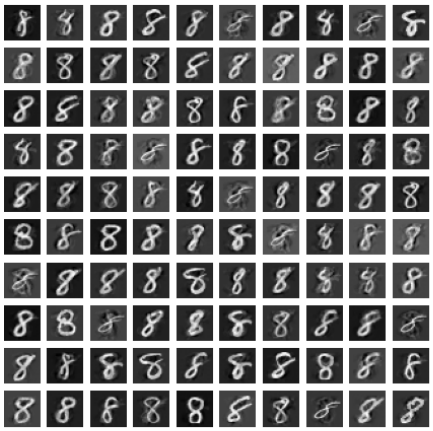


Figure 5.53: Samples generated by ILDM.

Then we give a quantitative measurement. Table 5.14 compares the performance of the ILDM and LDM models on the MNIST dataset using FID and LPIPS metrics, initialised with 300 points of random starting noise(results are averaged over 10 runs). The ILDM model, trained for only 500 epochs, consistently achieved better (lower) FID scores than the LDM model, which was trained for 10,000 epochs (177.25 vs. 182.69 at condition 3, and 148.91 vs. 167.17 at condition 9). For the LPIPS metric, ILDM showed a better score at condition 3 (0.5006 vs. 0.5813), while LDM achieved a marginally lower score at condition 9 (0.4870 vs. 0.5181). Overall, ILDM demonstrated superior or competitive performance despite significantly less training.

Metric	**ILDM**		**LDM**	
	3	**9**	**3**	**9**
MEAN±SD FID ↓	174±2.0	149±1.7	183±2.6	166±2.1
MEAN±SD LPIPS ↓	0.5006±0.01	0.5181±0.014	0.5813± 0.013	0.4870±0.01

Table 5.16: MNIST: FID comparison with random starting noise (300 points). LDM:10000 epochs; ILDM: 500 epochs; “nor” and “hcm” indicate respective categories.

5.5 Latent Diffusion Model Structure

To facilitate a fair and reproducible comparison between the proposed ILDM and baseline latent diffusion models, we summarise the neural architectures used to parameterise the score functions in our experiments. In all datasets, both LDM and ILDM employ a shared 1D U-Net–style ScoreNet backbone, with geometry-aware modifications applied only at

the level of diffusion dynamics and noise modeling. This section details the common ScoreNet architecture used across experiments, as well as its conditional extension, which serves as the backbone for all comparative evaluations.

Table 5.17: Overview of the LDM scorenet architecture. The network adopts a 1D U-Net–like encoder–decoder structure, where the diffusion time step is embedded via Gaussian Fourier features and injected into every layer through dense projections.

Module	Input Shape	Operation	Output Shape	Description
Input	(B, 1, L)	–	(B, 1, L)	1D input signal.
Time Embedding	(B, 1)	Gaussian Fourier Projection + Linear(256→256) + Swish ($x \cdot \sigma(x)$)	(B, 256)	Encodes the diffusion time step into a latent feature vector.
Encoder Block 1	(B, 1, L)	Conv1D(1→32, k=3, s=1, p=1, no bias) + Dense(256→32) + GroupNorm(4) + Swish	(B, 32, L)	Base-resolution feature extraction with time-conditioned modulation.
Encoder Block 2	(B, 32, L)	Conv1D(32→128, k=3, s=2, p=1, no bias) + Dense(256→128) + GroupNorm(32) + Swish	(B, 128, L/2)	Downsampling and feature refinement.
Encoder Block 3	(B, 128, L/2)	Conv1D(128→256, k=3, s=2, p=1, no bias) + Dense(256→256) + GroupNorm(32) + Swish	(B, 256, L/4)	Deep latent representation at the bottleneck.
Decoder Block 3	(B, 256, L/4)	ConvT1D(256→128, k=3, s=2, p=1, op=1, no bias) + Dense(256→128) + GroupNorm(32) + Swish	(B, 128, L/2)	First upsampling stage with time-conditioned normalisation.
Decoder Block 2	(B, 128+128, L/2)	ConvT1D(256→32, k=3, s=2, p=1, op=1, no bias) + Dense(256→32) + GroupNorm(32) + Swish	(B, 32, L)	Second upsampling stage with skip connection from Encoder Block 1. Interpolation is used if needed to align spatial length.
Decoder Block 1	(B, 32+32, L)	ConvT1D(64→1, k=3, s=1, p=1)	(B, 1, L)	Final reconstruction of the score field in the original signal space.
Normalisation	(B, 1, L)	Elementwise division by $\sigma(t)$ via `marginal_prob_std(t)`	(B, 1, L)	Time-dependent standard deviation normalisation of the output.

Table 5.17 provides an overview of the ScoreNet used in the latent diffusion model. The network adopts a 1D U-Net–style encoder–decoder design with skip connections, enabling the model to combine local detail with deeper latent features. The diffusion time step is encoded through Gaussian Fourier features and injected into every block via dense projections, allowing the representation at each layer to adapt to the current diffusion level. The encoder extracts progressively coarser features, while the decoder reconstructs the score estimate by merging upsamplled activations with corresponding encoder features. A final normalisation by the marginal standard deviation $\sigma(t)$ ensures that the predicted score matches the underlying SDE parameterisation.

Table 5.18: Overview of the **conditioned ScoreNet** architecture. The network extends the base 1D U-Net–style ScoreNet by incorporating an additional condition vector (e.g., angle) processed through a dedicated MLP. The resulting condition embedding is fused with the Gaussian Fourier time embedding and injected into all layers via dense projections.

Module	Input Shape	Operation	Output Shape	Description
Input	(B, 1, L)	–	(B, 1, L)	1D input signal.
Condition MLP	(B, 2)	Linear(2→256) + ReLU + Linear(256→256) + ReLU	(B, 256)	Encodes the angle or auxiliary condition into a latent vector.
Time Embedding	(B, 1)	Gaussian Fourier Projection + Linear(256) + Swish	(B, 256)	Fourier time embedding for diffusion step.
Fusion (Cond + Time)	(B, 256), (B, 256)	Elementwise addition	(B, 256)	Combined embedding injected into all network layers.
Encoder Block 1	(B, 1, L)	Conv1D(1→32) + Dense(256→32) + GroupNorm(4) + Swish	(B, 32, L)	First feature extraction layer.
Encoder Block 2	(B, 32, L)	Conv1D(32→128, stride=2) + Dense(256→128) + GroupNorm(32) + Swish	(B, 128, L/2)	Downsampling and deeper feature representation.
Encoder Block 3	(B, 128, L/2)	Conv1D(128→256, stride=2) + Dense(256→256) + GroupNorm(32) + Swish	(B, 256, L/4)	Bottleneck latent encoding.
Decoder Block 3	(B, 256, L/4)	ConvT1D(256→128, stride=2) + Dense(256→128) + GroupNorm(32) + Swish	(B, 128, L/2)	First upsampling stage.
Decoder Block 2	(B, 128+128, L/2)	ConvT1D(256→32, stride=2) + Dense(256→32) + GroupNorm(32) + Swish	(B, 32, L)	Second upsampling stage with skip from Encoder Block 1.
Decoder Block 1	(B, 32+32, L)	ConvT1D(64→1)	(B, 1, L)	Reconstruction of the score field.
Output Normalisation	(B, 1, L)	Divide by $\sigma(t)$ via `marginal_prob_std(t)`	(B, 1, L)	Ensures correctness under the diffusion SDE.

The conditioned ScoreNet extends the basic U-Net–style structure by adding a dedicated conditioning pathway. Alongside the Gaussian Fourier embedding used for the diffusion time, an auxiliary input (for example, an angle or other physical parameter) is passed through a two-layer MLP to obtain a matching 256-dimensional representation. This condition embedding is added to the time embedding to form a shared latent vector, which is injected into every encoder and decoder layer through small dense projections. In this way, the network's feature maps are modulated jointly by the time step and the external condition. The rest of the architecture follows a three-stage encoder–decoder design with skip connections and transposed convolutions for upsampling. The final output

is normalised by the marginal noise standard deviation to ensure consistency with the diffusion model.

Both the conditioned and unconditioned variants of the ScoreNet architecture are applied across all three datasets studied in this thesis. While the network backbone remains consistent, minor dataset-specific adjustments are made to the dimensionality of the input and output layers, as well as to the form of the conditioning variable when applicable. These modifications are limited to the interface components of the model; the encoder–decoder structure, embedding mechanism, and diffusion-based formulation are otherwise identical across datasets, ensuring methodological coherence throughout all experiments.

5.6 Conclusion

This chapter evaluated the proposed Intrinsic Latent Diffusion Model (ILDM) across three datasets with markedly different geometric structure and dimensionality: COIL-100 (Lucky Cat), left-ventricle cardiac MRI, and a small subset of MNIST. Using both qualitative comparisons and quantitative metrics (FID and LPIPS), ILDM consistently outperformed a standard pixel-space diffusion model (DM) and a Euclidean latent diffusion model (LDM). In particular, ILDM produced samples with clearer global structure, fewer distortions, and improved perceptual fidelity, while DM often failed to generate coherent structure and LDM tended to introduce artifacts when the latent geometry was not well approximated by a Euclidean space.

Beyond sample quality, latent-space analyses further supported the benefit of incorporating intrinsic geometry. The learned score fields and sampling trajectories under ILDM aligned more closely with the true latent data distribution, whereas LDM exhibited more convergent, geometry-mismatched vector fields and broader deviations of samplled points from the latent manifold. Ablation studies on the scaling parameter α showed that performance is sensitive to the thresholding of mapping uncertainty, with moderate values yielding the best FID across datasets, indicating a balance between geometric constraint and sampling flexibility.

Overall, these experiments demonstrate that enforcing intrinsic geometric structure in latent diffusion leads to more faithful score estimation and more reliable generation across diverse data regimes, including low-dimensional object manifolds, limited-sample handwritten digits, and highly structured medical imagery.

5.7 Limitations

A primary limitation lies in computational scalability. Simulating Brownian motion can be computationally expensive, particularly for large datasets or higher-dimensional latent spaces. Although the current implementation is suitable for moderate-scale problems, future work will focus on developing sparse approximations that enable the proposed method to scale to substantially larger datasets.

Another limitation arises from the geometric assumptions of the framework. The current approach relies on a single smooth latent chart learned by a GPLVM or variational autoencoder, which may be insufficient for real-world data exhibiting complex topology, multiple charts, or intersecting submanifolds. Extending the framework to atlas-based or multi-chart representations, and incorporating tools from topological data analysis, would allow more faithful modeling of such complex geometric structures.

A further limitation lies in the lack of theoretical understanding of the proposed hybrid Riemannian–Euclidean diffusion process. While the proposed hybrid Riemannian Euclidean diffusion process demonstrates strong empirical performance, its theoretical properties remain insufficiently understood. A rigorous analysis of long-term behavior, convergence guarantees, and principled criteria for selecting latent dimensionality—potentially based on geometric uncertainty, spectral properties of the learned metric, or information-theoretic measures—would further strengthen the theoretical foundations of the framework.

Chapter 6

Gaussian Process on an Unknown Manifold

This chapter investigates regression problems where predictors lie on an unknown low-dimensional Riemannian manifold embedded in a high-dimensional ambient space, building upon the geometric and stochastic foundations introduced in Chapter 2, particularly Brownian motion and the heat kernel. We observe pairs $\{(s_i, y_i)\}_{i=1}^n$, where $s_i \in \mathbb{R}^p$ are predictors and $y_i \in \mathbb{R}$ are response variables, and the predictors are assumed to lie on an unknown manifold $\mathcal{M}$. Our goal is to construct a Gaussian process (GP) regression model that is intrinsic to the underlying manifold, so that similarity between inputs reflects geodesic structure and curvature rather than ambient Euclidean distance.

To achieve this, we adopt a two-stage strategy. First, we use a generative dimension reduction model to represent the unknown data manifold. The model provides latent coordinates and an induced Riemannian metric (Section 2.2.2). To avoid unreliable extrapolation, we define an uncertainty-based boundary $\partial\mathcal{M}$ using the predictive variance of the mapping and restrict geometric computations to its interior. Second, based on this learned geometry, we construct a geometry-aware covariance kernel for f by approximating the heat kernel on the inferred manifold. We leverage the equivalence between the heat kernel and Brownian motion transition densities, and we also discuss a graph-Laplacian-based alternative. The resulting model: GPUM enables principled prediction and uncertainty quantification on point-cloud data without requiring explicit charts or known analytic geometry.

6.1 Problem Setup: Regression on an Unknown Manifold

We consider a regression problem in which the predictors lie on an unknown low-dimensional manifold embedded in a high-dimensional ambient space. Specifically, we observe pairs

$$(s_i, y_i), \qquad i = 1, \ldots, n,$$

where $y_i \in \mathbb{R}$ is a scalar response and $s_i = (s_i^1, \ldots, s_i^p) \in \mathbb{R}^p$ denotes the observed predictor. The response is assumed to follow the model

$$y_i = f(s_i) + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2), \tag{6.1}$$

with $f : \mathcal{M} \to \mathbb{R}$ an unknown regression function defined on a Riemannian manifold $M \subset \mathbb{R}^p$.

The manifold $\mathcal{M}$ is assumed to be smooth, compact, and q-dimensional with $q \ll p$. Although $\mathcal{M}$ is embedded in $\mathbb{R}^p$, its intrinsic geometry is governed by a local coordinate system $x \in \mathbb{R}^q$ through an unknown smooth embedding map

$$\phi : \mathbb{R}^q \to \mathbb{R}^p, \qquad s = \phi(x) + \eta, \qquad \eta \sim \mathcal{N}(0, \sigma_s^2 I).$$

Here, $s \in \mathbb{R}^p$ denotes the observed predictor introduced earlier, now modelled as a noisy observation of a latent point x. In classical settings, such intrinsic coordinates may be explicitly known. For example, points on the unit sphere in $\mathbb{R}^3$ admit a natural two-dimensional parameterisation via latitude and longitude. In many modern applications, however, only the ambient coordinates $\{s_i\}$ are observed, possibly corrupted by noise, while the intrinsic coordinates $\{x_i\}$ and the embedding map ϕ remain latent.

This setting is common in high-dimensional scientific data, including image analysis, computer vision, sensor networks, and biological data, where observations form a point cloud samplled from an unknown manifold, as illustrated in the heart dataset experiments in Chapter 5. In such cases, the intrinsic geometry–including geodesic distances, curvature, and volume distortion–plays a critical role in determining how information should be shared across observations. The central challenge is therefore to construct a regression model for f that accounts for this latent geometric structure, while relying only on the observed point cloud in the ambient space.

Why Euclidean Kernels Fail and Why the Heat Kernel

Standard Gaussian process regression typically employs covariance functions based on Euclidean distances between predictors [2]. When predictors are samplled from a nonlinear manifold embedded in a high-dimensional space, ambient Euclidean distance can be a poor proxy for intrinsic similarity. Points that are distant along the manifold may appear close in $\mathbb{R}^p$, causing Euclidean kernels to smooth the regression function across regions that are not intrinsically connected.

Applying Euclidean kernels in a learned latent space does not fully resolve this issue, since latent embeddings are generally non-isometric. Local distortions induced by curvature and sampling density lead to spatially varying smoothness, violating the stationarity assumptions underlying standard kernel designs [77] , which assume that the covariance depends only on the relative distance between points and is invariant across the space. These limitations motivate the use of covariance functions that are intrinsically defined on the manifold and adapt to its geometry.

6.2 Related Works

Several Gaussian process models on manifolds have been proposed under the assumption that the manifold geometry is fully known [23, 58, 74]. Niu et al. [23] constructed intrinsic Gaussian processes by defining heat kernels associated with the Laplace–Beltrami operator on Riemannian manifolds, explicitly incorporating boundary and interior conditions and yielding covariance structures that respect the intrinsic geometry. Lin et al. [58] proposed extrinsic GP models, where the manifold is embedded into a Euclidean space and Gaussian processes are defined using kernels based on ambient-space distances rather than intrinsic geometry. Related work by Dunson et al. [59], Borovitskiy et al. [60], and Bolin et al. [63] developed GP models on graphs and metric graphs induced by manifold-structured data. These approaches provide principled probabilistic models but rely critically on explicit geometric knowledge and are not applicable when only point cloud observations are available.

Graph Laplacian methods approximate differential operators on manifolds through nearest-neighbor or kernel-weighted graphs. Theoretical studies establish that properly normalised graph Laplacians converge to the Laplace–Beltrami operator under suitable sampling conditions [66, 68, 69]. These ideas underpin spectral clustering [70], graph-based semi-supervised learning [71], and diffusion-based regularisation frameworks. However, such methods require sufficiently dense sampling, exhibit sensitivity to bandwidth selection, and typically lack a probabilistic interpretation.

Several probabilistic manifold learning models have been proposed to learn low-

dimensional latent representations of high-dimensional data. Gaussian process latent variable models (GP-LVM) [77] and their deep variants define a generative mapping from a latent space to the ambient space and provide uncertainty quantification for the learned embedding. These models are well suited for nonlinear dimension reduction and data generation.

Overall, existing approaches either assume full knowledge of the manifold geometry, provide only deterministic embeddings without probabilistic modeling, or learn latent representations that do not capture intrinsic geometric structure. These limitations motivate a unified framework that jointly learns manifold geometry from data and constructs a geometry-aware GP model without requiring explicit charts.

Finally, it is worth emphasizing that existing geometry-aware GP models can be broadly categorised into deterministic-manifold methods and graph-based discrete approximations. These approaches implicitly assume a fixed and globally valid geometry, and typically do not model geometric uncertainty or the effective boundary of the data-supported manifold. In contrast, GPUM defines a Gaussian process prior directly on a probabilistic manifold geometry inferred from data, where both the Riemannian metric and the uncertainty-supported manifold interior are treated as random objects induced by the latent mapping.

6.3 Gaussian Process Prior

With the regression problem defined, we place a GP prior on f whose covariance is defined by the (estimated) heat kernel on the learned manifold geometry. The covariance kernel is central to this prior, as it encodes assumptions about smoothness and structure. Classical kernels such as the squared exponential and the Matérn kernel typically depend on the Euclidean distance $\|s_i - s_j\|$. While effective in Euclidean spaces, such kernels do not reflect the geometry of the underlying manifold when the predictors lie on a curved subset of $\mathbb{R}^p$. Consequently, they may produce incorrect similarity measures and degrade predictive performance when the data follow a nonlinear geometric structure.

The heat kernel provides a natural generalisation of radial basis kernels to Riemannian manifolds. As the fundamental solution to the heat equation on M, it is determined by the Laplace–Beltrami operator and captures the local diffusion geometry of the manifold, thereby incorporating geometric structure directly into the GP prior. However, its analytic form is rarely accessible except for a few special manifolds [17]. In general, computing the heat kernel requires solving partial differential equations on the manifold or employing spectral expansions, both of which become intractable when the manifold is unknown.

Niu et al. [23] propose numerical methods for estimating heat kernels when an explicit analytical parameterisation of the manifold is available. Their framework achieves accurate approximations but relies fundamentally on the known geometry. In high-dimensional point cloud settings, where no parameterisation is provided and the geometry must be inferred, these methods cannot be directly applied. This highlights the need for GP methodologies capable of leveraging manifold structures even in the absence of explicit charts.

6.4 Learning Geometry of Implicit Manifolds

This section instantiates the probabilistic geometry learning paradigm developed in Chapter 2 for intrinsic Gaussian process regression. We reuse the probabilistic latent mapping framework to infer an uncertainty-aware representation of the data manifold, including a prior over local Riemannian metrics, which together provide the geometric foundation of the intrinsic GP prior.

We now describe how to infer this implicit manifold geometry from an observed point cloud without requiring explicit charts. Specifically, probabilistic generative dimension reduction models are employed to obtain a latent coordinate representation that captures the local geometric structure, and to define an effective manifold boundary based on predictive uncertainty. Once this probabilistic geometry is learned, the corresponding heat kernel can be constructed or approximated on the inferred manifold.

Building on Riemannian geometry, an effective strategy is to simulate Brownian motion paths on the learned manifold (see Section 2.1.2) and estimate the heat kernel as the associated transition density. This approach circumvents the need for analytic parameterisations and enables data-driven construction of a manifold-aware kernel suitable for GP modeling. The resulting geometry-adaptive kernel allows the regression function f to be inferred even when the manifold structure is unknown.

Let

$$D = \{(s_i, y_i) : i = 1, \ldots, n\}$$

denote the labelled dataset and

$$V = \{s_{n+1}, \ldots, s_{n+v}\}$$

the unlabelled observations. Under the proposed GPUM (Gaussian Process on an Unknown Manifold) prior,

$$f \sim \mathrm{GP}(0, K_t^{\mathrm{heat}}(\cdot, \cdot)),$$

where K_t^{heat} is the heat kernel estimated on the inferred manifold geometry. The prior distribution of f given the labelled data is

$$p(f \mid s_1, \ldots, s_n) \sim N(0, \Sigma_{ff}),$$

with covariance matrix

$$\Sigma_{ff}^{i,j} = \sigma_h^2 K_t^{\text{heat}}(s_i, s_j).$$

Here, σ_h^2 is a scaling hyperparameter that provides additional flexibility in the covariance structure.

6.5 Boundary Definition and Uncertainty Quantification

In regions of the latent space where data are sparse or entirely absent, the GPLVM prior for the mapping function ϕ naturally yields larger predictive variance. Since the Riemannian metric on the learned manifold depends explicitly on the Jacobian of ϕ, uncertainty in the GP prior propagates to uncertainty in the induced geometry. As a result, geometric quantities such as distances, volumes, and curvature become unreliable in these regions.

To ensure that subsequent geometric operations-including metric evaluation and heat-kernel estimation are conducted only in regions where the learned geometry is trustworthy, we introduce an uncertainty-based boundary for the inferred manifold. Let

$$\text{Var}(\phi(x) \mid x)$$

denote the predictive variance of the GPLVM at a latent point x. We define the uncertainty-based boundary of the learned manifold as

$$\partial\mathcal{M} = \{x \in \mathbb{R}^q \mid \text{Var}(\phi(x) \mid x) = \alpha\}, \tag{6.2}$$

where α is chosen as the maximum predictive variance observed across the latent coordinates of the training set. Points satisfying $\text{Var}(\phi(x) \mid x) > \alpha$ correspond to regions where the GP prior is forced to extrapolate and where both the mapping and its Jacobian are poorly constrained. Within the boundary, the manifold geometry induced by the GP mapping is supported by the observed data and is therefore considered reliable. This uncertainty-based boundary is subsequently used to constrain Brownian motion (BM) paths for heat-kernel estimation. In particular, when BM reaches $\partial\mathcal{M}$, we enforce a boundary-handling strategy to keep trajectories inside the well-supported region, ensuring

stable and accurate estimation of transition densities near the boundary. A detailed comparison between reflection and resampling-based boundary treatments and their impact on heat-kernel accuracy is provided in [67].

All subsequent geometric computations, including heat kernel approximation and Brownian motion simulation, are restricted to the well-supported region enclosed by $\partial\mathcal{M}$, with explicit boundary-handling strategies applied when trajectories reach the boundary to ensure numerical stability and geometric fidelity.

6.6 Heat Kernel Estimation I: Brownian Motion Transition Densities

We approximate the heat kernel under the boundary-constrained Brownian motion to construct the GP covariance. Our primary estimator exploits the equivalence between the heat kernel and Brownian motion transition densities, enabling a simulation-based, coordinate-independent approximation.

Specifically, on a Riemannian manifold, the transition density of Brownian motion coincides with the heat kernel associated with the Laplace–Beltrami operator [15]. This follows from the fact that Brownian motion is the diffusion process whose infinitesimal generator is one half of the Laplace–Beltrami operator, so that its stochastic differential equation can be derived from the Laplace–Beltrami operator, and its transition density is given by the fundamental solution of the corresponding heat equation [18].

Consider a Brownian motion $\{S(t) \mid t > 0\}$ evolving on a manifold $\mathcal{M} \subset \mathbb{R}^p$. The approach relies on simulating multiple sample paths of the process, typically generated via numerical integration of the stochastic differential equations introduced in Subsection 2.1.2 [18, 19]. To begin, we initiate N independent Brownian motion paths, each starting at the same point $S(0) = s_0 \in \mathcal{M}$ at time $t = 0$. Our goal is to estimate the transition density of Brownian motion at a target point $s \in \mathcal{M}$ at time t, given the initial condition $S(0) = s_0$. Since the analytical form of the transition probability is unavailable, the transition density cannot be obtained by directly differentiating $P\{S(t) \in A_s \mid S(0) = s_0\}$. To facilitate this, we define a small neighborhood of the target point s, denoted as $A_s \subset \mathcal{M}$. This neighborhood represents a region around the target point where we wish to evaluate the probability of path arrival.

For any time $t > 0$, the probability that a Brownian path, originating at s_0, reaches the neighborhood A_s can be estimated by counting the number of simulated Brownian motion paths that intersect A_s at time t. This is a direct application of the Monte Carlo

method for probability estimation [20]. This leads to an approximation of the transition probability, represented as:

$$P\{S(t) \in A_s \mid S(0) = s_0\} \approx \frac{k}{N}, \tag{6.3}$$

where k is the number of simulated Brownian motion sample paths that reach the neighborhood A_s at time t. N is the total number of simulated Brownian motion sample paths. This estimated probability, k/N, represents the proportion of paths that have diffused from the starting point into the region A_s. Significantly, this transition probability corresponds to the integral of the Brownian motion transition density over the neighborhood A_s [21]. Since the transition density is equivalent to the heat kernel (as established in Subsection 2.1.2), we can approximate the heat kernel at the target point s based on the ratio of reaching sample paths within the neighborhood. Therefore, the transition density of $S(t)$ at s, often denoted as the heat kernel $K_{\text{heat}}(s_0, s, t)$, can be approximated as:

$$K_{\text{heat}}(s_0, s, t) \approx \frac{P\{S(t) \in A_s \mid S(0) = s_0\}}{V(A_s)} \approx \frac{1}{V(A_s)} \frac{k}{N} = \hat{K}_{\text{heat}}, \tag{6.4}$$

where $V(A_s)$ denotes the Riemannian volume of the neighborhood A_s. The volume is parameterised by its radius. As shown in [23], increasing $V(A_s)$ reduces Monte Carlo sampling error but enlarges numerical approximation error, and an optimal neighborhood size can be chosen by minimizing their combined effect.

The Monte Carlo estimation procedure described above is illustrated in Fig. 6.1. In this example, three independent Brownian motion paths are simulated on a Swiss roll manifold, all starting from the same initial point s_0. A small neighborhood A_s is defined around a target point s. Among the three sample paths, only one trajectory enters A_s at time t, yielding an empirical transition probability of $1/3$. This example visualises how the ratio k/N serves as an unbiased estimator of the probability mass of the Brownian transition density over A_s, since each simulated BM path independently enters A_s with probability $P\{S(t) \in A_s \mid S(0) = s_0\}$, yielding

$$\mathbb{E}\left[\frac{k}{N}\right] = P\{S(t) \in A_s \mid S(0) = s_0\}.$$

Thereby BM path provides an intuitive interpretation of the heat kernel estimator in eq (6.4).

The accuracy of this approximation is directly influenced by the size of the neighborhood A_s and, consequently, its volume. Choosing an appropriate neighborhood size is

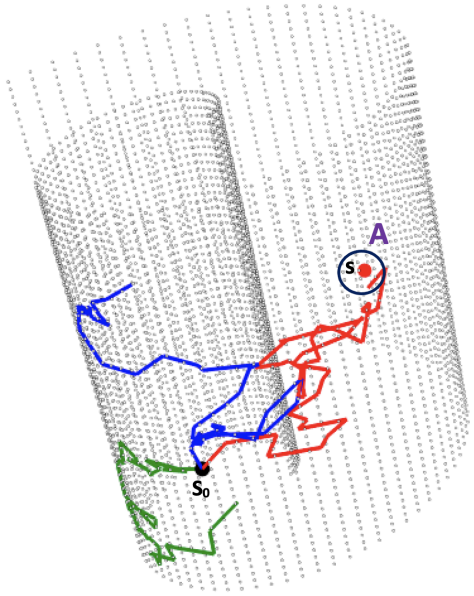


Figure 6.1: Three BM paths (red, blue, green) are simulated on the Swiss roll, with the starting point at s_0 (black ball). s (red ball) is the target point. A is the neighbourhood of s. Only the red path reaches A at time t. The transition probability $p\{S(t) \in A \mid S(0) = s_0\}$ is 1/3. Adapted from [30].

crucial; too small and the probability estimate may be susceptible to statistical noise (due to insufficient k), while too large and the approximation may not be representative of the density at the target point s (as it averages over too wide a region) [22]. A critical property of this transition density estimator is its coordinate independence. The stochastic differential equation in (2.10) is obtained by expressing the heat equation associated with the Laplace–Beltrami operator in local coordinates as a Fokker-Planck equation, which implies that the resulting Brownian motion and its transition density are invariant under smooth reparameterisations of the latent space. Consequently, the estimated heat kernel depends only on the intrinsic Riemannian geometry of the manifold and is independent of the choice of local coordinate system or parameterisation [18].

The above Monte Carlo estimation procedure naturally leads to a practical algorithm for constructing the intrinsic GP covariance matrix from simulated Brownian motion paths. Algorithm 6 summarises the implementation details for estimating the heat kernel K^t_{heat} and the corresponding covariance matrix Σ^t at a given diffusion time t.

Algorithm 6 Estimating K^t_{heat} and Σ^t from BM simulations

Estimate the heat kernel and covariance matrix.
Given a discrete choice of diffusion time $t \in \{\Delta t, 2\Delta t, \ldots, N_t \Delta t\}$, the covariance matrix Σ^t is estimated using the BM sample paths generated in Algorithm 1.1.

for $i = 1, \ldots, n$ **do**
  for $j = 1, \ldots, n$ **do**
    $N_{\mathbf{A}_j} \leftarrow \text{which}\big(x(t) \in \mathbf{A}_j\big)$ {count BM paths reaching $\mathbf{A}_j$}
    $K^t_{heat}(s_i, s_j) = \dfrac{N_{\mathbf{A}_j}}{N_{BM}\, V(\mathbf{A}_j)}$ {N_{BM} is No. of trajectories, use eqn (2.8)}
    $\Sigma^t_{ij} = \sigma_h^2\, K^t_{heat}(s_i, s_j)$
  end for
end for
return Σ^t

In our experiments, $N_{BM} = 20000$ BM paths are simulated.

6.7 Heat Kernel Estimation II: Graph Laplacian Approximation (Baseline)

As an alternative and computational baseline, we approximate the heat kernel via spectral truncation using a graph Laplacian constructed on the observed point cloud. This approach leverages the classical spectral representation of the heat kernel in terms of the eigenpairs of the Laplace–Beltrami operator.

Let $\mathcal{M}$ be a d-dimensional smooth, compact, and connected Riemannian manifold embedded in $\mathbb{R}^p$. Denote by $-\Delta$ the Laplace–Beltrami operator on $\mathcal{M}$, with eigenvalues $0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq \cdots$ and corresponding orthonormal eigenfunctions $\{\phi_i\}_{i=0}^{\infty}$, satisfying $-\Delta\phi_i = \lambda_i \phi_i$ and forming an orthonormal basis in $L^2(\mathcal{M})$. The heat kernel admits the classical spectral expansion [9, 10]:

$$H(x, x', t) = \sum_{i=0}^{\infty} e^{-\lambda_i t}\, \phi_i(x)\phi_i(x'). \tag{6.5}$$

In practice, when the manifold structure is unknown and only a collection of samplled points is available, direct computation of the Laplace–Beltrami eigenpairs is not feasible. A commonly used surrogate is to approximate the operator $-\Delta$ through a graph Laplacian constructed on the data cloud. Under appropriate sampling assumptions and bandwidth choices, graph Laplacians are known to converge to the Laplace–Beltrami operator in both spectral and operator norms (e.g. [64, 65, 69]). Consequently, the eigenvalues and eigenvectors of a suitably normalised graph Laplacian can be used as substitutes for $\{\lambda_i, \phi_i\}$,

where λ_i and ϕ_i denote the eigenvalues and eigenvectors of the discrete graph Laplacian, respectively, yielding an approximation of the heat kernel:

$$H(x, x', t) = \sum_{i=0}^{\infty} e^{-\mu_i t}\, \tilde{v}_i(x)\tilde{v}_i(x'),$$

where μ_i and $\tilde{v}_i$ denote the eigenvalues and eigenvectors of the graph Laplacian.

Graph construction. Given a dataset

$$X := \{x_1, x_2, \ldots, x_n, \cdots, x_{n+v}\}, \quad x_i \in \mathbb{R}^p,$$

with n labelled points and v unlabelled or auxiliary grid points, we follow the normalised graph Laplacian construction described in Dunson et al. (2020). We begin by defining the Gaussian-like kernel

$$k_\varepsilon(x, x') = \exp\left(-\frac{\|x - x'\|^2_{\mathbb{R}^p}}{4\varepsilon^2}\right),$$

with bandwidth parameter $\varepsilon > 0$. The affinity matrix W is constructed using the α-normalised kernel, where

$$q_\varepsilon(x) := \sum_{i=1}^{n} k_\varepsilon(x, x_i);$$

$$W_{ij} := \frac{k_\varepsilon(x_i, x_j)}{q_\varepsilon(x_i)q_\varepsilon(x_j)} = \frac{k_\varepsilon(x_i, x_j)}{\sum_{i=1}^{n+v} k_\varepsilon(x_i, x_j) \sum_{j=1}^{n+v} k_\varepsilon(x_i, x_j)}.$$

A diagonal degree matrix D is defined by

$$D_{ii} = \sum_{j=1}^{n+v} W_{ij}.$$

The associated Markov transition matrix is

$$A = D^{-1}W.$$

The graph Laplacian estimator is then given by

$$L := \frac{A - I}{\varepsilon^2}.$$

For numerical convenience, we also define the symmetric matrix

$$\tilde{A} = D^{-1/2} W D^{-1/2},$$

which shares the same eigenvalues as A and is diagonally orthogonalizable.

Spectral approximation and normalisation. Let $\mu_{i,\varepsilon}$ denote the i-th eigenvalue of $-L$, and $\tilde{v}_{i,\varepsilon}$ its corresponding eigenvector, normalised in ℓ^2. To estimate the continuum normalisation of eigenfunctions, we follow the density-corrected normalisation of Dunson et al. (2020). Define

$$N(i) = |B_\varepsilon^{\mathbb{R}^p}(f(x_i)) \cap \{f(x_1) \cdots f(x_n)\}|,$$

the number of sample points within an ε-ball around x_i. Then the discretised ℓ^2 norm is

$$\|\tilde{v}\|_2 = \sqrt{\frac{|S^{d-1}|\varepsilon^d}{d} \sum_{i=1}^{n} \frac{\tilde{v}^2(i)}{N(i)}}.$$

The normalised eigenvector is

$$v_{i,n,\varepsilon} := \frac{\tilde{v}_{i,\varepsilon}}{\|\tilde{v}\|_2}.$$

Graph Laplacian heat kernel estimator Using the first K eigenpairs, the heat kernel approximation becomes

$$\text{GLkernel}_t = \sum_{i=0}^{K-1} e^{-\mu_{i,\varepsilon} t}\, v_{i,\varepsilon} v_{i,\varepsilon}^T,$$

where K determines the truncation level of the spectral expansion. In practice, K is chosen to balance approximation accuracy and computational cost, typically selected to capture the dominant low-frequency eigenmodes. In our experiments, we use $K \in [5, 10]$, which was found to provide a good trade-off. This construction parallels the classical continuum expansion and provides a computationally tractable estimator of the heat kernel when only point cloud data are observed.

The complete procedure is summarised in Algorithm 7.

Algorithm 7 GL Algorithm

Require: t, ε, K

1: **Step (1):** Construct the $(n+v)\times(n+v)$ matrix W and D with bandwidth ε and cloud points $\{x_1,\dots,x_{n+v}\}$. Compute:

$$\tilde{A} = D^{-1/2} W D^{-1/2}.$$

2: **Step (2):** Find the first $K-1$ eigenpairs of $\tilde{A}$:

$$\{\alpha_{i,\varepsilon}, U_{i,\varepsilon}\}_{i=1}^{K-1}.$$

3: **Step (3):** Suppose $\tilde{v}_{i,\varepsilon}$ is the normalised vector of $D^{-1/2}U_{i,\varepsilon}$ in the ℓ^2 norm. Compute

$$\mu_{i,\varepsilon} := \frac{1-\alpha_{i,\varepsilon}}{\varepsilon^2}.$$

Let

$$N(i) = |B_\varepsilon^{\mathbb{R}^p}(f(x_i)) \cap \{f(x_1),\dots,f(x_n)\}|.$$

Then the ℓ^2 norm of $\tilde{v}$ is

$$\tilde{v}_2 = \sqrt{\frac{|S^{d-1}|\varepsilon^d}{d}\sum_{i=1}^{n}\frac{\tilde{v}^2(i)}{N(i)}}.$$

For $i = 1, 2, \dots, K-1$, set:

$$v_{i,\varepsilon} := \frac{\tilde{v}_{i,\varepsilon}}{\tilde{v}_2}.$$

4: Construct $H_{\varepsilon,t}^K$ as

$$H_{\varepsilon,t}^K = \sum_{i=0}^{K-1} e^{-\mu_{i,\varepsilon} t}\, v_{i,\varepsilon} v_{i,\varepsilon}^T.$$

6.8 Experiments

We evaluate GPUM on both synthetic and real dataset to assess whether the proposed geometry-aware kernel improves prediction accuracy relative to Euclidean and graph-based baselines. The experiments are designed to examine the ability of the method to exploit the intrinsic geometry of the underlying manifold in regression tasks.

We consider a synthetic Swiss roll example and a real-world image dataset. The Swiss roll simulation provides a controlled setting in which the true manifold geometry and regression function are known, enabling detailed qualitative and quantitative comparisons. The COIL image experiment demonstrates the applicability of the proposed approach to high-dimensional data, where the manifold structure must be inferred from observations. Across all experiments, we report both visualisations and predictive error

metrics to compare the proposed method with competing approaches.

6.8.1 Simulation Study on Swiss Roll

In this section, we conduct a simulation study to evaluate the performance of the proposed method on a regression task defined over the Swiss roll, a two-dimensional manifold embedded in $\mathbb{R}^3$. The point cloud representation of the Swiss roll is shown in Fig. 6.2, consisting of $n = 24$ labelled points and $v = 450$ unlabelled points. The labels are generated from a known regression function defined on the manifold, serving as ground truth for evaluation. Both labelled and unlabelled observations are used in B-GPLVM to learn the latent representation of the manifold. The learned latent space, visualised as an unfolded surface in Fig. 6.3, displays the unlabelled points as blue triangles and labelled points as red dots. The predictive variance of the GP mapping ϕ is plotted as the gray background. As expected, regions distant from observed points exhibit higher variance, indicated by darker shading.

Using the definition in eq (6.2), the boundary $\partial\mathcal{M}$ of the learned manifold is shown in Fig. 6.5. The black regions on the left and right lie outside $\partial\mathcal{M}$, while the central white region corresponds to the interior where geometric computations are considered reliable. The magnification factor is displayed in Fig. 6.4. Consistent with Fig. 2.3, the horizontal axis corresponds to the scaled radius of the Swiss roll. Larger radii correspond to larger magnification factors (darker colors). Due to fewer observations near the tail of the roll, the B-GPLVM geometry becomes less accurate in this region, as evidenced by the rightmost column of latent points being separated from the main cluster in Fig. 6.5. Once the metric $\mathcal{G}$ and boundary $\partial\mathcal{M}$ are obtained, the heat kernel can be estimated by simulating Brownian motion paths on the implicit manifold.

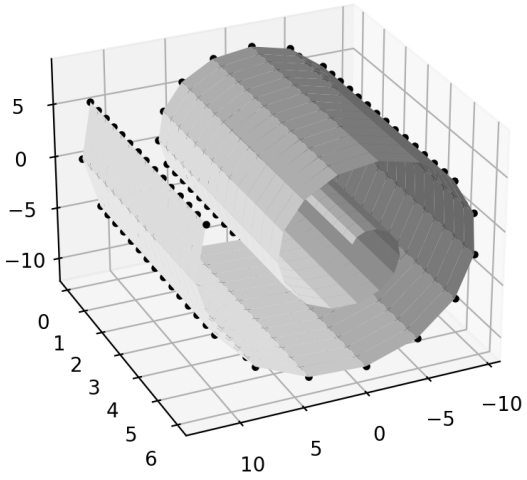


Figure 6.2: Swiss roll point cloud in $\mathbb{R}^3$.

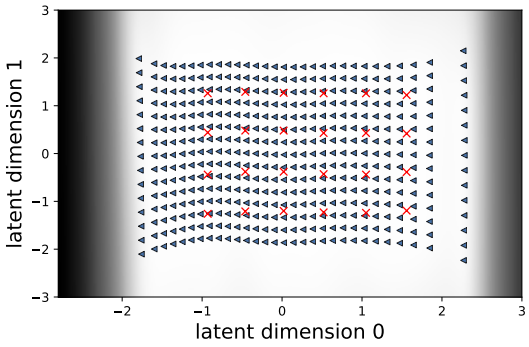


Figure 6.3: Latent space learned by Bayesian GPLVM with background grayscale indicating mapping variance.

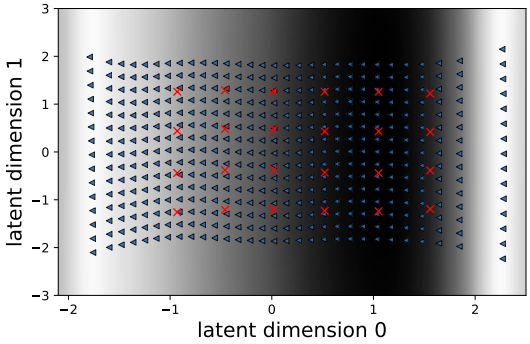


Figure 6.4: Latent space visualisation with magnification factor; darker colors correspond to larger magnification.

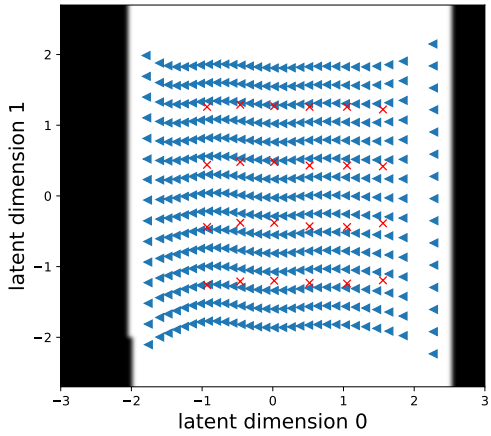


Figure 6.5: Latent space with boundary visualisation, where the dark region lies outside $\partial\mathcal{M}$ and the white region lies inside.

For the labelled points, the response variables are

$$y_i = f(s_i^1, s_i^2, s_i^3) + \epsilon_i, \quad i = 1 \dots n, \quad s_i \in \mathbb{R}^3 \tag{6.6}$$

where f is the unknown regression function and s_i denotes the embedded coordinates of points on $\mathcal{M}$. For visualisation purposes, the true regression function is plotted on the unfolded Swiss roll in Fig. 6.6, using an analytical two-dimensional parameterisation (radius and width). The 24 labelled observations are marked with black crosses. The true function varies smoothly for small radii but changes more rapidly for larger radii.

We first apply a standard Euclidean GP in $\mathbb{R}^3$ (as described in [2]), using a squared

exponential kernel with Euclidean distance. This model ignores the intrinsic structure of the Swiss roll and allows interactions between points that are close in ambient space but distant along the manifold. The predictive mean for the unlabelled points is shown in Fig. 6.7. Compared with the true function in Fig. 6.6, the prediction contours are overly wiggly, and the color patterns deviate significantly, particularly in regions with radius 6, 8, and 10.

As a second baseline, we consider a GP defined in the two-dimensional latent space obtained from B-GPLVM, using the Euclidean distance in the squared exponential kernel, denoted $\mathbb{R}^2$ GP. This approach ignores the manifold geometry, including the learned metric and magnification factors. The predictive mean is shown in Fig. 6.10. Since the regression function is highly nonstationary in the latent space, the $\mathbb{R}^2$ GP severely underfits the data.

The predictive mean obtained from the proposed GPUM model is shown in Fig. 6.8. Its overall structure closely matches the true function, both in the shape of level sets and in color patterns. The GL-GP prediction with $K = 10$ eigenfunctions is shown in Fig. 6.11. Because the GL kernel estimates in Fig. 6.9 deviate substantially from the analytical heat kernel, the GL-GP predictions are also poor. A one-dimensional comparison is provided in Fig. 6.9, where predictions are plotted along the curve defined by $z = 4$ while varying the radius from 2 to 12.5. The $\mathbb{R}^2$ GP (brown dotted line) underfits, while the $\mathbb{R}^3$ GP (green dashed line) oscillates in the opposite direction from the ground truth (blue solid line). The GL-GP (black dashed line) also oscillates around the true curve. In contrast, the GPUM prediction (red dashed line) most accurately follows the overall pattern of the true function.

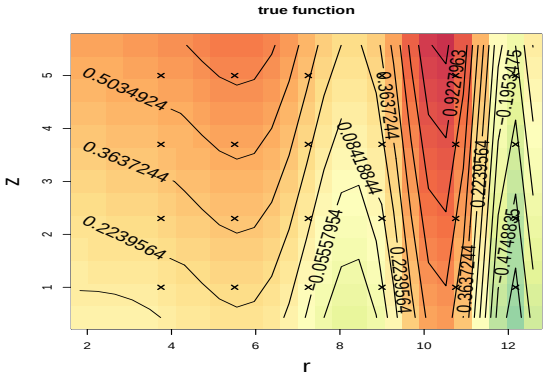


Figure 6.6: True function on the unfolded Swiss roll, with labelled data points marked by black crosses.

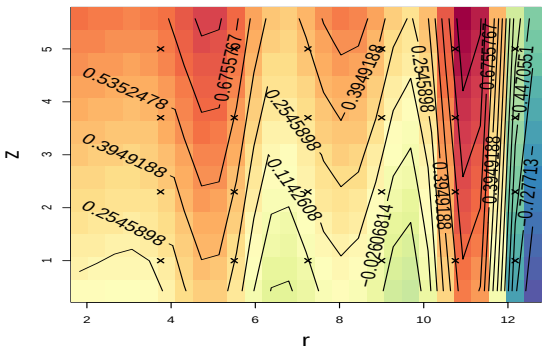


Figure 6.7: Prediction using a Euclidean GP constructed in $\mathbb{R}^3$.

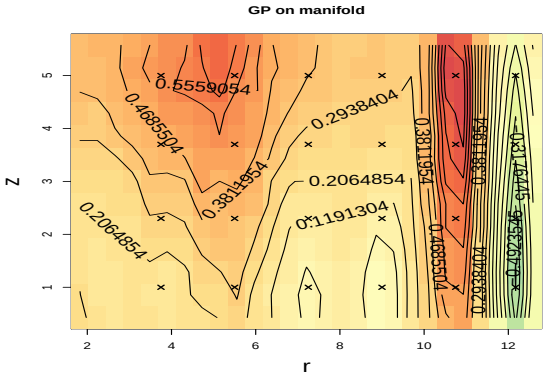


Figure 6.8: Prediction using the proposed intrinsic Gaussian process (IGP).

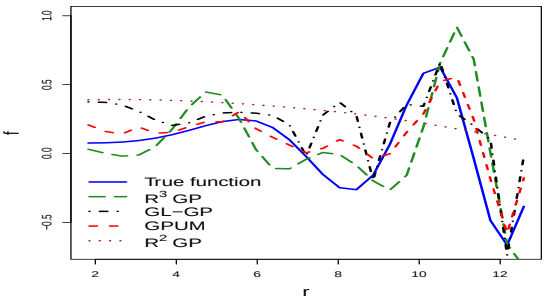


Figure 6.9: One-dimensional comparison of predictions obtained by fixing $z = 4$ and varying the radius from 2 to 12.5.

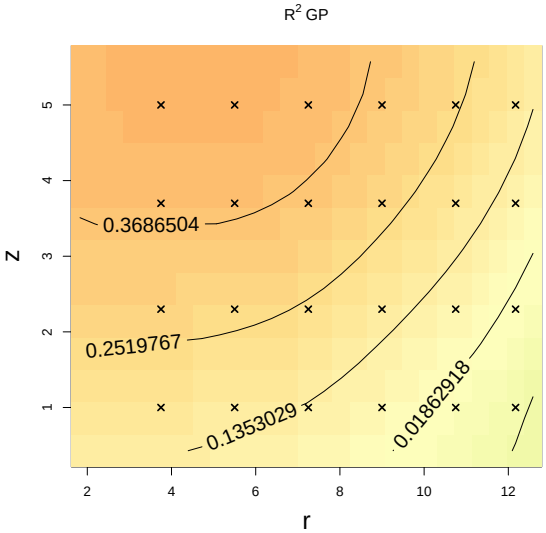


Figure 6.10: Prediction using a Euclidean GP constructed in $\mathbb{R}^2$.

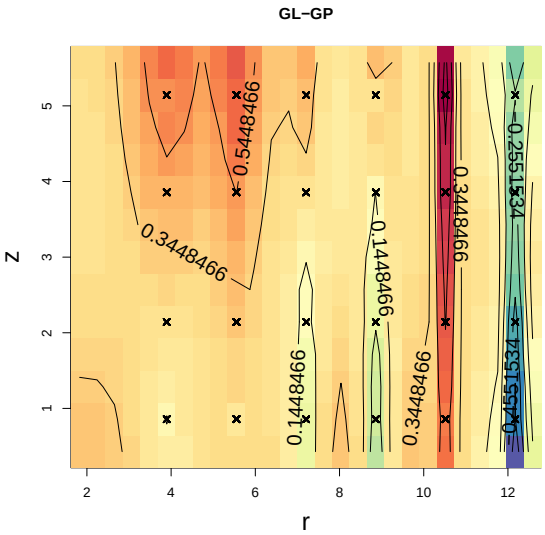


Figure 6.11: Prediction using the graph Laplacian Gaussian process baseline.

To evaluate performance under different sampling regimes, we consider three scenarios with varying numbers of unlabelled grid points ($v = 250$, 450, and 800). These points are used to estimate the manifold geometry for GPUM , GM-GP, and GL-GP, and also serve as the test sets for evaluating regression accuracy. For each scenario, we generate 20 training sets by randomly selecting $n = 23$ labelled points. All five models (GPUM , GM-GP, GL-GP, $\mathbb{R}^2$ GP, and $\mathbb{R}^3$ GP) produce predictions at the test locations. The root mean squared error (RMSE) between predictive means and true function values is computed, and the mean and standard deviation over the 20 training sets are reported in Table 6.1.

Table 6.1: Comparison of the root mean squared errors of five methods on Swiss roll. Values in parentheses show the standard deviation of the RMSE over 20 independently generated training sets.

	$\mathbb{R}^3$ GP	$\mathbb{R}^2$ GP	GPUM	GL-GP	GM-GP
mean RMSE $v = 250$	0.284±0.006	0.293±0.005	0.163±0.020	0.243±0.003	0.231±0.001
mean RMSE $v = 450$	0.298±0.007	0.290±0.005	0.162±0.003	0.220±0.002	0.207±0.002
mean RMSE $v = 800$	0.287±0.006	0.282±0.005	0.164±0.002	0.216±0.001	0.206±0.001

The results show strong and consistent performance of GPUM across all three sampling scenarios. GPUM outperforms all competing methods by a substantial margin. GL-GP and GM-GP display similar RMSE values and benefit from denser sampling on the manifold, performing better when more grid points are available. Both graph-based models significantly outperform the Euclidean GPs. Notably, GPUM achieves the lowest RMSE even with fewer grid points, demonstrating its efficiency and robustness relative to approaches that rely solely on graph approximations. Both GPUM and GL-GP also provide predictive uncertainty through the posterior variance of the Gaussian process. In practice, higher uncertainty is typically observed in regions with sparse or noisy observations. A separate uncertainty quantification comparison was not included because several baseline methods do not provide uncertainty estimates in a directly comparable way. Therefore, Tables 6.1 and 6.2 focus primarily on predictive accuracy metrics.

6.8.2 Wifi Signal

Indoor positioning is an important application, yet conventional localisation methods such as the Global Positioning System (GPS) are unreliable in indoor environments. Instead, indoor localisation can be achieved by exploiting wireless signals emitted by devices such as WiFi access points. In this section, we study the problem of estimating two-dimensional indoor locations from WiFi signal strength measurements.We use a dataset collected by Ferris et al. [72], where a mobile device traverses a single floor of a university building while recording signal strength traces from 30 WiFi access points. Since collecting labelled location data is often costly, we simulate a low-label regime by assuming that only three

locations are known and aim to infer the positions of all remaining measurements from their corresponding WiFi signal observations.

This localisation task can be formulated as a regression problem, where the position of the mobile device is modelled as a function of the observed WiFi signal strengths. Let $s_i \in \mathbb{R}^{30}$ denote the vector of signal measurements from the 30 access points at the i-th location, and let y_i be the corresponding coordinate of the device. The relationship is written as

$$y_i = f(s_i) + \varepsilon_i, \qquad i = 1, \ldots, n,$$

where f is an unknown regression function and ε_i represents measurement noise.

In total, WiFi signals are recorded at $n + v = 36$ locations. Only $n = 3$ of these locations are associated with known position coordinates, while the remaining measurements are unlabell ed. To prevent the three labelled points from being spatially clustered, we select them from three distinct regions whose union covers the entire dataset. The remaining $v = 33$ locations are treated as unlabelled test points, on which different methods are evaluated for coordinate estimation. This random partitioning is repeated 20 times to generate multiple training and testing splits.

Unlike the Swiss roll simulation, the WiFi measurements lie in a high-dimensional space that cannot be directly visualised, and the structure of the underlying manifold is unknown. We therefore first learn a two-dimensional latent representation of the data using B-GPLVM, which serves as a chart for the inferred manifold. The latent dimensionality is selected as $q = 2$ based on the ARD weights, which quantify the contribution of each latent dimension. The latent space is plotted with the variance of the mapping as the gray background in Figure 6.12. The 36 WiFi signal strength measurements are represented by the blue triangles. The training set (the labelled points) is marked by the red crosses. The dark color value is for high uncertainty. As the mobile device traverses a looped trajectory, the inferred latent representation forms a closed curve in the latent space. The boundary of the implicit manifold, $\partial\mathcal{M}$, defined by Eqn (6.2), is shown in Fig 6.13. The magnification factor is visualised as the gray-scale background in Fig 6.14, where darker shades indicate larger magnification values. A sample Brownian motion trajectory on the inferred manifold is illustrated in Fig 6.15. Under the Neumann boundary condition, the Brownian motion paths are constrained to remain within the manifold boundary.

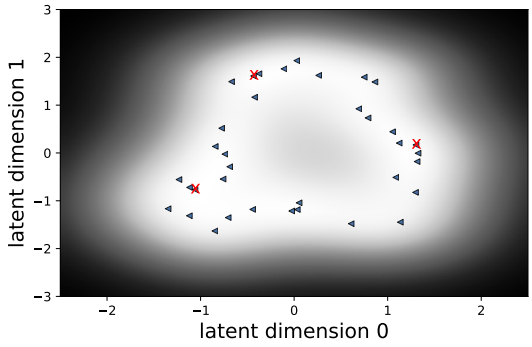


Figure 6.12: Latent space constructed by Bayesian GPLVM. Blue triangles indicate unlabelled points and red crosses indicate labelled points. Darker background colors represent higher predictive variance of the mapping.

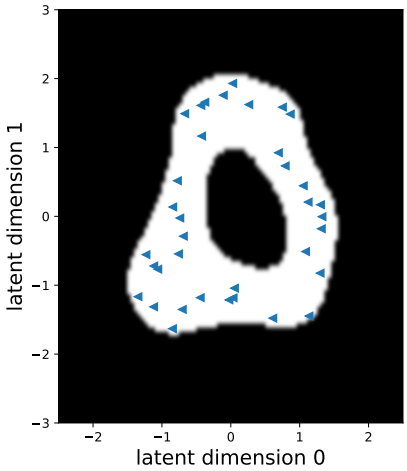


Figure 6.13: Latent space highlighting the uncertainty-based boundary $\partial\mathcal{M}$. Dark regions are outside the boundary and white regions are inside the well-supported domain.

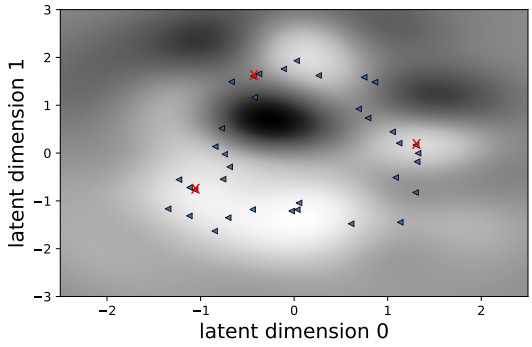


Figure 6.14: Latent space visualisation with magnification factor; darker colors correspond to larger magnification values.

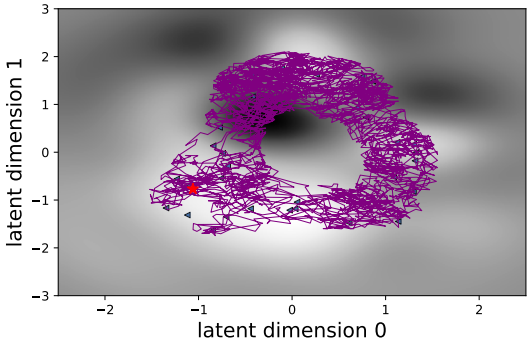


Figure 6.15: A sample Brownian motion trajectory on the inferred manifold. The red star denotes the starting location.

The coordinates of the mobile device are estimated using five different methods. The ground-truth locations are indicated by distinct colors in the latent space shown in Fig 6.16. We then apply a Gaussian process defined in $\mathbb{R}^{30}$ using a squared exponential kernel with Euclidean distances between WiFi signal vectors, thereby ignoring the intrinsic manifold structure of the data. The predictive means of this $\mathbb{R}^{30}$ GP are shown in Fig 6.17. In contrast, the predictive mean of GPUM, shown in Fig 6.18, exhibits an overall spatial pattern that is much closer to the ground truth. As summarised in Table 6.2, GPUM attains the lowest average RMSE on the WiFi dataset, while also exhibiting relatively small variability across different random splits.

Table 6.2: Comparison of the root mean squared errors of five methods on WiFi signal data. Values in parentheses denote the standard deviation of the RMSE across different random training splits.

	$\mathbb{R}^{30}$GP	$\mathbb{R}^{2}$GP	GPUM	GL-GP	GM-GP
MEAN RMSE	5.57±1.43	4.83±1.83	4.11±0.88	5.6±1.15	6.04±0.57

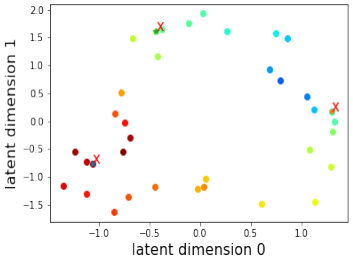


Figure 6.16: Ground truth in the latent space.

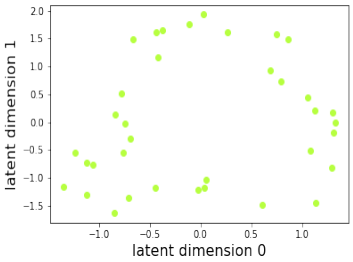


Figure 6.17: $\mathbb{R}^{30}$ GP prediction.

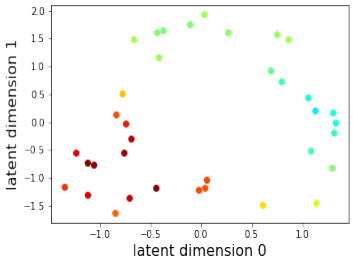


Figure 6.18: GPUM prediction.

6.9 Conclusion

Across both synthetic and real-world experiment, the proposed GPUM consistently demonstrates improved predictive performance over Euclidean and graph-based baselines. The Swiss roll study highlights the importance of respecting intrinsic geometry, showing that ignoring manifold structure leads to inappropriate smoothing and distorted predictions even when a latent representation is available. In contrast, the geometry-aware kernel employed by GPUM accurately captures the underlying variation of the regression function.

The wifi signal experiment further illustrates the practical relevance of the proposed approach in high-dimensional settings where the manifold structure is unknown and must be inferred from data. By incorporating learned geometry and restricting diffusion to data-supported regions, GPUM achieves robust predictions without relying on explicit parameterisations or dense sampling. Together, these results demonstrate that explicitly modeling intrinsic geometry is crucial for Gaussian process regression on manifold-valued data, and that the proposed framework provides an effective and flexible solution in both controlled and real-world scenarios.

GPUM share the same limitations with ILDM: they face scalability challenges due to the cost of simulating Brownian motion in high-dimensional settings, and they rely on a single smooth latent chart, which may be inadequate for data with complex or multi-chart geometric structure.

The quality of the learned geometry is also sensitive to the pretrained latent representation. Inaccuracies in the decoder-induced geometry or regions of poor latent support

may propagate into the diffusion dynamics, potentially affecting the stability and quality of generated samples. Leveraging high-quality pretrained latent models, such as externally trained variational autoencoders, offers a promising direction to improve geometric estimation, reduce training cost, and facilitate transfer across related datasets.

Chapter 7

Conclusion and Future Research

7.1 Summary of Contributions

This thesis develops probabilistic learning methods for data supported on unknown manifolds, with the aim of explicitly incorporating intrinsic geometry into both regression and generative modeling. Rather than relying on Euclidean assumptions or purely local approximations, the work seeks to infer geometric structure from data and to use this structure directly in probabilistic inference.

The first main contribution is the Intrinsic Hybrid Latent Diffusion Model for generative modeling. In this framework, the latent space is interpreted as a chart of an unknown manifold equipped with a geometry induced by a pretrained decoder. Diffusion dynamics are defined in a hybrid manner: Riemannian Brownian motion is used in regions where the geometry is well supported by data, while Euclidean diffusion is applied in regions of high uncertainty. This leads to stochastic dynamics that adapt to spatially varying latent geometry and produce generative trajectories that better follow the structure of the data manifold.

A further contribution is the development of a practical training procedure for such hybrid diffusion processes. An approximate denoising score matching objective is used to learn the associated score functions without requiring closed-form transition densities, and a corresponding reverse-time sampling procedure is derived by combining Riemannian and Euclidean Langevin dynamics.

The second contribution of this thesis is the development of an intrinsic Gaussian process framework for regression on unknown manifolds. A probabilistic latent representation of the data manifold is learned using Gaussian Process Latent Variable Models, which induces a Riemannian metric and an associated measure of geometric uncertainty.

Brownian motion is then simulated on the inferred manifold and used to estimate the corresponding heat kernel, which serves as the covariance function of the Gaussian process. This construction avoids explicitly computing manifold geometry while providing regression models whose predictions and uncertainty estimates remain consistent with the intrinsic geometry of the data.

The effectiveness of the proposed methods is demonstrated through experiments on synthetic examples and on real-world datasets, including image and medical imaging data. Across these settings, the geometry-aware models consistently yield improved predictive performance and more coherent generative samples compared to Euclidean and graph-based baselines.

7.2 Final Remarks

This thesis investigated probabilistic learning on data supported by unknown nonlinear manifolds, with the aim of incorporating intrinsic geometric structure into both regression and generative modeling. Rather than treating geometry as an implicit byproduct of representation learning, the work explicitly infers geometric structure from data and uses it directly in stochastic modeling.

The results show that geometry-aware modeling leads to more consistent predictions, more meaningful uncertainty estimates, and more coherent generative behaviour across a range of datasets. The intrinsic Gaussian process and the intrinsic hybrid latent diffusion model provide concrete examples of how geometric information can be integrated into probabilistic learning in a practical and effective manner.

Appendix A

Swiss Roll Parameterisation

The three-dimensional coordinates of the Swiss Roll can be parametrised by the radius r and the width z. Consider the Swiss roll parametrised by

$$\mathbf{x}(r, z) = (r\cos r,\ r\sin r,\ z).$$

To find its metric tensor, we first compute the partial derivatives

$$\mathbf{x}_r = (\cos r - r\sin r,\ \sin r + r\cos r,\ 0), \quad \mathbf{x}_z = (0, 0, 1).$$

The metric tensor is given by

$$(\mathbf{x}_r \cdot \mathbf{x}_r)dr^2 + 2(\mathbf{x}_r \cdot \mathbf{x}_z)dr\,dz + (\mathbf{x}_z \cdot \mathbf{x}_z)dz^2 = (1 + r^2)dr^2 + dz^2.$$

or in matrix form

$$g = \begin{bmatrix} 1 + r^2 & 0 \\ 0 & 1 \end{bmatrix}, \qquad g^{-1} = \begin{bmatrix} \frac{1}{1+r^2} & 0 \\ 0 & 1 \end{bmatrix}, \qquad \frac{\partial g}{\partial r} = \begin{bmatrix} 2r & 0 \\ 0 & 0 \end{bmatrix}.$$

The determinant of the metric tensor in this case would be $1 + r^2$, as r grows the determinant is getting bigger. This indicates the exaggeration from the low dimensional latent space to the high dimensional original space is getting bigger.

The BM on the Swiss Roll can be written as

$$\begin{aligned} dr(t) &= -\frac{1}{2}\frac{r}{(1 + r^2)^2}dt + (1 + r^2)^{-1/2}dB_r(t), \\ dz(t) &= dB_z(t). \end{aligned} \tag{A.1}$$

Appendix B

Autoencoder

While Gaussian Process mappings offer a probabilistic route for learning local geometry from data, a parallel line of research has turned to neural network architectures capable of learning low-dimensional structure directly from high-dimensional samples. Among these, autoencoders constitute one of the most widely used and conceptually straightforward approaches. They provide a deterministic, neural-network-based alternative to the GP latent mappings introduced in the previous section, and they are particularly effective when nonlinear structure must be captured from large-scale datasets. This section introduces the basic formulation and theoretical background of autoencoders, which later serve as a point of comparison with probabilistic manifold models.

Autoencoders have been first introduced in 1986 [85] as a simple neural network that is trained to reconstruct the input. The problem is to learn the functions $A : \mathbb{R}^n \to \mathbb{R}^p$ (encoder) and $B : \mathbb{R}^p \to \mathbb{R}^n$ (decoder) that satisfy [86]:

$$arg\ min_{A,B} E[\Delta(\mathbf{x}, B \circ A(x))] \tag{B.1}$$

where E is the expectation over the distribution of x, and Δ is the reconstruction loss function, which measures the distance between the output of decoder and the input. A concise definition, adapted from Bank et al. (2020), is as follows:

Definition B.1. *An autoencoder is a model whose primary objective is to learn an informative, low-dimensional representation of the data by training a neural network to reconstruct its input observations with minimal error.*

B.1 Theory

Before examining specific neural architectures, we outline the conceptual structure common to all autoencoders. At its core, an autoencoder is an unsupervised neural network designed to perform dimensionality reduction and representation learning. It does so by compressing the input into a latent representation and then reconstructing the input from this compressed form. Autoencoders have been widely deployed for feature extraction, anomaly detection, denoising, and, more recently, as building blocks for generative models [31].

The architecture consists of three principal components:

- Encoder: The encoder transforms the input data into a lower-dimensional representation, referred to as the "latent space" or "bottleneck layer." This encoding captures the salient features of the input data, effectively mapping the input to a more compact representation. The latent space typically has a dimensionality lower than the original input, thereby enabling efficient data representation by reducing redundancy and noise while preserving critical information. The architecture of an encoder can vary depending on the specific task and model requirements. A common implementation in neural networks uses a feedforward architecture. This architecture consists of multiple layers of interconnected neurons. Each neuron calculates a weighted sum of its inputs, followed by a non-linear activation function (e.g., ReLU, sigmoid, or tanh) to introduce non-linearity into the model. These layers are connected sequentially, and the final layer outputs the encoded representation.

- Latent space: This is the core of the network, where the data is represented in a compressed, lower-dimensional space. The latent space aims to capture the most essential, underlying features of the input data in a compact form, serving as a meaningful abstraction of the original data. The dimensionality of the latent space is a critical hyperparameter that influences the model's ability to reconstruct the input and its ability to generalise to unseen data.

- Decoder: The decoder takes the compressed representation from the latent space and reconstructs the original data as accurately as possible. The decoder's architecture is generally the inverse of the encoder's, mirroring its layers and operations. The decoder aims to reverse the encoding process, mapping the latent space representation back to the original data space. Like the encoder, the decoder can utilise various neural network architectures, including feedforward networks, convolutional networks, or recurrent networks, depending on the nature of the data and the task at hand.

- Output Layer: The final layer of the decoder produces the output, which represents the autoencoder's reconstruction of the original input data. The nature of the output layer (e.g., linear, sigmoid, or softmax) depends on the type of data being reconstructed. For example, if the input data consists of real-valued numbers, a linear output layer might be appropriate. If the input data consists of probabilities, a sigmoid or softmax output layer might be used.

During training, the autoencoder aims to minimize the reconstruction error, which quantifies the discrepancy between the original input and the reconstructed output. Common loss functions used to achieve this minimisation include mean squared error (MSE), binary cross-entropy (for binary inputs), and variations thereof ([31]). The choice of loss function depends on the nature of the input data and the desired reconstruction characteristics. Regularisation techniques, such as L1 or L2 regularisation, are often employed to prevent overfitting and encourage the encoder to learn more robust and generalisable representations [32].

Overall, autoencoders provide a versatile framework for unsupervised learning, dimensionality reduction, and representation learning, enabling efficient data compression, anomaly detection, feature extraction, data denoising, and generative modeling in a wide range of domains.

B.2 Activation and Loss Functions in Neural Networks

Activation functions introduce nonlinearity into the model and are essential for capturing complex data structure. Common choices include:

- **ReLU:**
$$\mathrm{ReLU}(x) = \max(0, x),$$
widely used due to its simplicity and resistance to vanishing gradients.

- **SELU:** A self-normalizing activation defined as
$$\mathrm{SELU}(x) = \lambda \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0, \end{cases}$$
promoting stable activations in deep networks.

- **Sigmoid:**
$$\sigma(x) = \frac{1}{1 + e^{-x}},$$

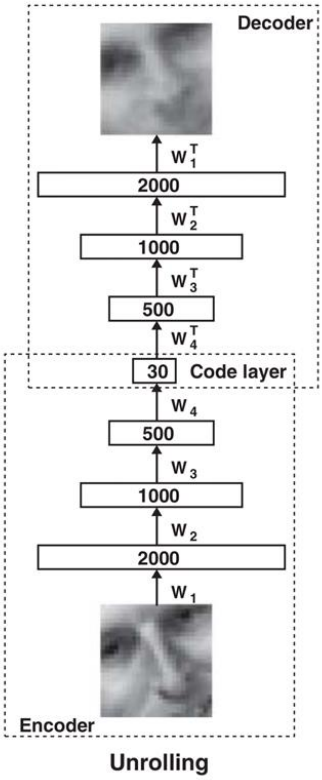


Figure B.1: Schematic architecture of a standard autoencoder.

mapping values to $(0, 1)$, often used when the reconstructed output represents probabilities.

Loss functions quantify reconstruction quality. Widely used examples include:

- **Mean Squared Error (MSE):**

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2,$$

simple but sensitive to outliers.

- **Reconstruction loss:**

$$L_{\text{recon}} = \frac{1}{N}\sum_{i=1}^{N}\|x_i - y_i\|^2,$$

commonly used when inputs are continuous images or signals.

- **Binary Cross-Entropy (BCE):**

$$L_{\text{BCE}} = -\sum_{i=1}^{n}\left[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)\right],$$

appropriate for binary or normalised pixel data.

Autoencoders thus provide a flexible framework for learning nonlinear representations from data. Their deterministic nature contrasts with the probabilistic GP mapping models introduced earlier, and the two families of methods offer complementary strengths. Neural-network-based autoencoders will serve as a comparative reference point when evaluating manifold learning methods in later chapters.

B.3 Auto-Encoder Metrics

Once we have constructed an Auto-encoder model and established a relatively robust latent space, we can proceed to determine the estimated heat kernel of the dataset (point cloud). The auto-encoder model could be seen as the probabilistic mapping function ϕ between the real dataset and the latent space.

In contrast to Gaussian–process–based mappings, the autoencoder provides a deterministic reconstruction map

$$\phi_\theta : \mathbb{R}^q \to \mathbb{R}^p,$$

parameterised by neural network weights θ. Once the autoencoder has been trained, the decoder ϕ_θ defines a smooth nonlinear embedding of the latent space into the ambient observation space. As in the previous sections, this mapping induces a Riemannian metric on the latent manifold via the pullback of the Euclidean metric.

For any latent coordinate $x \in \mathbb{R}^q$, the Jacobian of the decoder is

$$J_\phi(x) = \frac{\partial \phi_\theta(x)}{\partial x} \in \mathbb{R}^{p\times q}.$$

Since ϕ_θ is a deterministic neural network, $J_\phi(x)$ is also deterministic and can be computed by automatic differentiation.

The Riemannian metric induced by the decoder is the pullback of the Euclidean metric in $\mathbb{R}^p$:

$$g_{\mathrm{AE}}(x) = J_\phi(x)^\top J_\phi(x), \tag{B.2}$$

which is a positive–semidefinite $q \times q$ matrix. This metric captures how local perturbations in the latent space are stretched by the nonlinear decoder when mapped to the observation space.

Unlike the GPLVM and Bayesian–GPLVM metrics developed in Sections 2.2.2 and 6.5, the autoencoder metric contains no epistemic uncertainty term. All geometric quan-

tities derived from g_{AE}—including geodesics, volume elements, and Laplace–Beltrami operators—are fully determined by the trained decoder. If stochastic regularisation such as dropout is used during inference, one may instead define an approximate stochastic metric via Monte Carlo sampling, but this is not required in the deterministic formulation adopted in this thesis.

Appendix C

Bayesian GPLVM Metric

Following the description in Section 3.2, the marginal likelihood can be derived from the augmented joint probability as

$$p(\mathcal{S}) = \int\int\int \prod_{j=1}^{p} p(s^j \mid \phi^j)\, p(\phi^j \mid u^j, \mathcal{X})\, p(u^j)\, p(\mathcal{X})\, d\mathcal{U}\, d\Phi\, d\mathcal{X}. \tag{C.1}$$

where $\mathcal{S} = \{s_i \mid i = 1, \ldots, n+v\}$, $s_i \in \mathbb{R}^p$, is the set of the observed data points in the original space. $\mathcal{X} = \{x_i \mid i = 1, \ldots, n+v\}$, $x_i \in \mathbb{R}^q$ is the set of latent (unobserved) variables. The mapping is denoted as $\Phi = \{\phi_i \mid i = 1, \ldots, n+v\}$, $\Phi \in \mathbb{R}^{(n+v)\times p}$ and $\phi_i^j = \phi(x_i)^j$. The inducing points are denoted by $\mathcal{U} = \{u_i \mid i = 1, \ldots, m\}$, $\mathcal{U} \in \mathbb{R}^{m\times p}$ and $u_i = \phi(x_{u_i}) \in \mathbb{R}^p$. The inducing points are evaluated at the pseudo-inputs $\mathcal{X}_u = \{x_{u_i} \mid i = 1, \ldots, m\}$, $\mathcal{X}_u \in \mathbb{R}^{m\times q}$, in the latent space.

$$p(\phi^j \mid u^j, \mathcal{X}, \mathcal{X}_u) = \mathcal{N}\left(\phi^j \mid K_{\mathcal{X},\mathcal{X}_u} K_{\mathcal{X}_u,\mathcal{X}_u}^{-1} u^j, K_{\mathcal{X},\mathcal{X}} - K_{\mathcal{X},\mathcal{X}_u} K_{\mathcal{X}_u,\mathcal{X}_u}^{-1} K_{\mathcal{X}_u,\mathcal{X}}\right), \tag{C.2}$$

$$p(u^j) = \mathcal{N}(u^j \mid 0, K_{\mathcal{X}_u,\mathcal{X}_u}). \tag{C.3}$$

We can now apply variational inference to approximate the true posterior,

$$p(\Phi, \mathcal{U}, \mathcal{X} \mid \mathcal{S}) = p(\Phi \mid \mathcal{U}, \mathcal{S}, \mathcal{X}) p(\mathcal{U} \mid \mathcal{S}, \mathcal{X}) p(\mathcal{X} \mid \mathcal{S}), \tag{C.4}$$

with a variational distribution of the form

$$q(\Phi,\mathcal{U},\mathcal{X}) = p(\Phi \mid \mathcal{U},\mathcal{X})q(\mathcal{U})q(\mathcal{X}) = \left(\prod_{j=1}^{p} p(\phi^j \mid u^j,\mathcal{X})q(u^j)\right) q(\mathcal{X}). \tag{C.5}$$

The distribution $q(\mathcal{X})$ is chosen to be Gaussian with variational parameters for mean and variance. Using this variational distribution and Jensen's inequality, we can derive the variational lower bound $\mathcal{F}$ of $\log p(\mathcal{S})$, the log marginal likelihood. $p(\mathcal{X})$ is chosen as a Gaussian prior with identity covariance. The particular choice for the variational distribution allows us to analytically compute a lower bound.

$$\mathcal{F}(q(\mathcal{X})q(\mathcal{U})) = \int q(\Phi,\mathcal{U},\mathcal{X}) \log \frac{p(\mathcal{S},\Phi,\mathcal{U},\mathcal{X})}{q(\Phi,\mathcal{U},\mathcal{X})}\, d\mathcal{X}\, d\Phi\, d\mathcal{U} = \hat{\mathcal{F}}(q(\mathcal{X}),q(\mathcal{U})) - \mathrm{KL}(q(\mathcal{X})\|p(\mathcal{X})). \tag{C.6}$$

Clearly, the second KL term in above can be easily calculated since both $p(\mathcal{X})$ and $q(\mathcal{X})$ are Gaussian. The first term can be written as

$$\hat{\mathcal{F}}(q(\mathcal{X}),q(\mathcal{U})) = \sum_{j=1}^{p} \hat{\mathcal{F}}^j(q(\mathcal{X}),q(\mathcal{U})), \tag{C.7}$$

$$\hat{\mathcal{F}}^j(q(\mathcal{X}),q(\mathcal{U})) = \int q(u^j) \log \frac{\exp\left(\log \mathcal{N}\left(s^j \mid K_{\mathcal{X},\mathcal{X}_u}K_{\mathcal{X}_u,\mathcal{X}_u}^{-1}u^j, \beta^{-1}I_{n+v}\right)\right)_{q(\mathcal{X})} p(u^j)}{q(u^j)}\, du^j + \mathcal{T}, \tag{C.8}$$

where

$$\mathcal{T} = \frac{\beta}{2}\mathrm{Tr}\left(\langle K_{\mathcal{X}\mathcal{X}}\rangle_{q(\mathcal{X})}\right) - \frac{\beta}{2}\mathrm{Tr}\left(K_{\mathcal{X}_u,\mathcal{X}_u}^{-1}\langle K_{\mathcal{X}_u,\mathcal{X}}K_{\mathcal{X},\mathcal{X}_u}\rangle_{q(\mathcal{X})}\right). \tag{C.9}$$

$\langle\cdot\rangle_{q(\mathcal{X})}$ denotes expectation under the distribution $q(\mathcal{X})$. The expression in the above equation is a KL-like quantity. And $q(u^j)$ is optimally set to be proportional to the numerator inside the logarithm, which is also a Gaussian distribution. The final expression for $\hat{\mathcal{F}}$ becomes

$$\hat{\mathcal{F}}(q(\mathcal{X})) = \sum_{j=1}^{p}\left(\log\left(\int \exp\left(\left\langle \log \mathcal{N}\left(s^j \mid K_{\mathcal{X},\mathcal{X}_u}K_{\mathcal{X}_u,\mathcal{X}_u}^{-1}u^j, \beta^{-1}I_{n+v}\right)\right\rangle_{q(\mathcal{X})}\right) p(u^j)\, du^j\right)\right) - \frac{\beta}{2}\mathrm{Tr}\left(\langle K_{\mathcal{X}\mathcal{X}}\rangle_{q(\mathcal{X})}\right) + \frac{\beta}{2}\mathrm{Tr}\left(K_{\mathcal{X}_u,\mathcal{X}_u}^{-1}\langle K_{\mathcal{X}_u,\mathcal{X}}K_{\mathcal{X},\mathcal{X}_u}\rangle_{q(\mathcal{X})}\right). \tag{C.10}$$

Bibliography

[1] Riemann, B. (1854). *Über die Hypothesen, welche der Geometrie zu Grunde liegen*. In: *Gesammelte Mathematische Werke und Wissenschaftlicher Nachlass*, pp. 272–287. Leipzig: Teubner.

[2] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press.

[3] Arvanitidis, G., Hansen, L. K., and Hauberg, S. (2018). *Latent space oddity: On the curvature of deep generative models*. In: *Proceedings of the International Conference on Learning Representations (ICLR)*.

[4] Arvanitidis, G., Hansen, L. K., and Hauberg, S. (2019). *Geometrically principled connections in latent space*. In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1369–1380.

[5] Titsias, M. (2009). *Variational learning of inducing variables in sparse Gaussian processes*. In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 567–574.

[6] do Carmo, M. P. (1992). *Riemannian Geometry*. Boston: Birkhäuser.

[7] Jost, J. (2017). *Riemannian Geometry and Geometric Analysis* (7th ed.). Cham: Springer.

[8] J. M. Lee, *Introduction to Smooth Manifolds*, Graduate Texts in Mathematics, Vol. 218, Springer, 2013.

[9] Rosenberg, S. (1997). *The Laplacian on a Riemannian Manifold: An Introduction to Analysis on Manifolds*. Cambridge: Cambridge University Press.

[10] Chavel, I. (1984). *Eigenvalues in Riemannian Geometry*. New York: Academic Press.

[11] Taylor, M. E. (2011). *Partial differential equations I: Basic theory*. Springer Science & Business Media.

[12] Evans, L. C. (2010). *Partial differential equations*. American Mathematical Society.

[13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *arXiv preprint arXiv:1406.2661*, 2014.

[14] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *arXiv preprint arXiv:1701.07875*, 2017.

[15] Lawler, G. F. (2010). *Brownian motion*. American Mathematical Society.

[16] Lawrence, N. D. (2004). *Local distance preservation in the Gaussian process latent variable model*. Advances in neural information processing systems, 17.

[17] Hsu, E. P. (1988). *Heat semigroup on a Riemannian manifold*. In Seminar of Probability XXII, 140-161. Springer, Berlin, Heidelberg.

[18] Hsu, E. P. (2008). *Stochastic Analysis on Manifolds*. Vol. 38 of Graduate Studies in Mathematics. American Mathematical Society, Providence, RI. (Reprint of the 2002 original)

[19] Milstein, G. N., Tretyakov, M. V. (2013). *Numerical Integration of Stochastic Differential Equations*. Berlin: Springer.

[20] Hammersley, J. M.,& Handscomb, D. C. (1964). *Monte Carlo Methods*. Methuen & Co Ltd, London.

[21] Billingsley, P. (1995). *Probability and Measure* (3rd ed.). New York: John Wiley & Sons.

[22] Rubinstein, R. Y., & Kroese, D. P. (2016). *Simulation and the Monte Carlo Method* (3rd ed.). Hoboken, NJ: John Wiley & Sons.

[23] Niu, M., Cheung, P., Lin, L., Dai, Z., Lawrence, N., and Dunson, D. (2019). *Intrinsic Gaussian processes on complex constrained domains*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 81(3), 603–627. https://doi.org/10.1111/rssb.12320

[24] Roweis, S. T., & Saul, L. K. (2000). *Nonlinear dimensionality reduction by locally linear embedding*. Science, 290(5500), 2323–2326.

[25] Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). *A global geometric framework for nonlinear dimensionality reduction*. Science, 290(5500), 2319–2323.

[26] Tosi, N., Ek, C. H., Solin, A., & Lawrence, N. D. (2014). *Metrics for Probabilistic Geometric Embeddings*. Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS), Proceedings of Machine Learning Research (PMLR), 33, 1042–1050.

[27] Damianou, A. C. (2016). *Deep Gaussian processes and variational propagation of uncertainty*. PhD thesis, University of Sheffield.

[28] Titsias, M. K. (2010). *Bayesian Gaussian process latent variable model*. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 844–851.

[29] Lawrence, N. D. (2007). *Learning for larger datasets with the Gaussian process latent variable model*. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 243–250.

[30] Niu, M., Dai, Z., Cheung, P., and Wang, Y. (2023). *Intrinsic Gaussian Process on Unknown Manifolds with Probabilistic Metrics*. Journal of Machine Learning Research, 24:1–42.

[31] Bank, D., Koenigstein, N.(2021). *Autoencoders*. arXiv: 2003.05991 [cs.LG].

[32] Goodfellow, I., Bengio, Y. (2016). *Deep Learning*. Cambridge, MA: MIT Press. http://www.deeplearningbook.org.

[33] Dubey, S. R., Singh, S. K., Chaudhuri, B. B. (2022). *Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark*. arXiv:2109.14545 [cs.LG].

[34] Terven, J., Cordova-Esparza, D. M., Ramirez-Pedraza, A., Chavez-Urbiola, E. A. (2023). *Loss Functions and Metrics in Deep Learning*. arXiv:2307.02694 [cs.LG].

[35] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B. (2022). High-Resolution Image Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10684-10695).

[36] Song, Y., & Ermon, S. (2019). *Generative modeling by estimating gradients of the data distribution*. Advances in Neural Information Processing Systems (NeurIPS), 32.

[37] Song, Y., Ermon, S. (2021). Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*.

[38] Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., & Ganguli, S. (2015). *Deep unsupervised learning using nonequilibrium thermodynamics*. International Conference on Machine Learning (ICML).

[39] Ho, J., Jain, A., & Abbeel, P. (2020). *Denoising Diffusion Probabilistic Models*. Advances in Neural Information Processing Systems (NeurIPS).

[40] Dhariwal, P., & Nichol, A. (2021). *Diffusion models beat GANs on image synthesis*. Advances in Neural Information Processing Systems (NeurIPS), 34.

[41] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, M., Ermon, S., & Poole, B. (2020). *Score-based generative modeling through stochastic differential equations*. International Conference on Learning Representations (ICLR).

[42] Vincent, P. (2011). *A connection between score matching and denoising autoencoders*. Neural Computation, 23(7), 1661–1674.

[43] Girolami, M., & Calderhead, B. (2011). *Riemann manifold Langevin and Hamiltonian Monte Carlo methods*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73(2), 123–214.

[44] Chen, T., Fox, E., & Guestrin, C. (2014). *Stochastic gradient Riemannian Langevin dynamics on the probability simplex*. In *Advances in Neural Information Processing Systems (NeurIPS)*, 27.

[45] Ho, J., & Salimans, T. (2022). *Classifier-free diffusion guidance*. Advances in Neural Information Processing Systems (NeurIPS), 35.

[46] Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI).

[47] Kingma, D. P., & Welling, M. (2019). *An introduction to variational autoencoders*. Foundations and Trends in Machine Learning, 12(4), 307–392.

[48] Lawrence, N. D. (2005). *Probabilistic non-linear principal component analysis with Gaussian process latent variable models*. Journal of Machine Learning Research (JMLR), 6, 1783–1816.

[49] Kloeden, P. E., & Platen, E. (1992). *Numerical Solution of Stochastic Differential Equations*. Springer, Berlin, Heidelberg.

[50] Lamberton, D., & Lapeyre, B. (2007). *Introduction to Stochastic Calculus Applied to Finance* (2nd ed.). Chapman and Hall/CRC.

[51] Nene, S. A., Nayar, S. K., & Murase, H. (1996). *Columbia Object Image Library (COIL-20)*. Technical Report CUCS-005-96, Columbia University.

[52] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition. Proceedings of the IEEE*, 86(11), 2278–2324.

[53] Bernard, O., Lalande, A., Zotti, C., Cervenansky, F., Yang, X., Heng, P.-A., Cetin, I., Lekadir, K., Camara, O., Ballester, M. A. G., Sanroma, G., Napel, S., Petersen, S. E., Petersen, M., et al. (2018). *Deep learning techniques for automatic MRI cardiac multi-structures segmentation and diagnosis: The ACDC challenge. Medical Image Analysis*, 43, 1–13.

[54] Campello, V. M., Gkontra, P., Izquierdo, C., Lekadir, K., & Camara, O. (2020). *Multi-centre, multi-vendor and multi-disease cardiac segmentation: The M&Ms challenge. Medical Image Analysis*, 70, 101918.

[55] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). *GANs trained by a two time-scale update rule converge to a local Nash equilibrium.* In *Advances in Neural Information Processing Systems (NeurIPS)*, 30.

[56] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., & Wang, O. (2018). *The unreasonable effectiveness of deep features as a perceptual metric.* In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595.

[57] Pope, P., Zhu, C., Abdelkader, A., Goldblum, M., & Goldstein, T. (2021). *The intrinsic dimension of images and its impact on learning.* In: Proceedings of the Ninth International Conference on Learning Representations (ICLR 2021).

[58] Lin, L., Dunson, D. B., and Carin, L. (2019). Extrinsic Gaussian processes for manifold-valued data. *Journal of Machine Learning Research*, 20(101), 1–44.

[59] Dunson, D. B., Wu, H.-T., and Wu, Y. (2020). Multiscale Gaussian processes on graphs and manifolds. *Journal of the American Statistical Association*, 115(531), 1327–1341.

[60] Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. P. (2021). Matérn Gaussian processes on Riemannian manifolds. *Journal of Machine Learning Research*, 22(60), 1–45.

[61] E. Mathieu, T. Rainforth, N. Siddharth, and Y. W. Teh, Riemannian Hamiltonian Monte Carlo on Learned Manifolds, in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[62] C. Fefferman, S. Mitter, and H. Narayanan, Testing the Manifold Hypothesis, *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

[63] Bolin, D., Kirchner, K., and Kovács, M. (2022). Stochastic partial differential equation approximations for Gaussian processes on metric graphs. *Bernoulli*, 28(4), 2673–2701.

[64] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[65] R. R. Coifman and S. Lafon, "Diffusion maps," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 5–30, 2006.

[66] M. Belkin and P. Niyogi, "Convergence of Laplacian eigenmaps," *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 129–136, 2007.

[67] Yuan Liu. *Sparse Intrinsic Gaussian Processes in Complex Constrained Domains with Application of Bayesian Optimisation.* PhD thesis, University of Glasgow, 2025.

[68] A. Singer, "From graph to manifold Laplacian: The convergence rate," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 128–134, 2006.

[69] N. G. Trillos and D. Slepčev, "Error estimates for spectral convergence of the graph Laplacian on random geometric graphs," *SIAM Journal on Mathematical Analysis*, vol. 52, no. 6, pp. 5594–5630, 2020.

[70] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 849–856, 2002.

[71] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 912–919, 2003.

[72] B. Ferris, D. Fox, and N. Lawrence, "WiFi-SLAM Using Gaussian Process Latent Variable Models," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[73] R. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete structures," *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 315–322, 2002.

[74] V. Borovitskiy, A. Terenin, P. Mostowsky, M. Deisenroth, and A. Wilson, "Matern Gaussian processes on Riemannian manifolds," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[75] A. Feragen, F. Lauze, and M. Nielsen, "Geodesic exponential kernels: When curvature and linearity conflict," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 10, pp. 1736–1750, 2015.

[76] K. Liang and B. Nadler, "Heat kernel regression on manifolds," *Journal of Machine Learning Research*, vol. 23, pp. 1–45, 2022.

[77] N. D. Lawrence, "Gaussian process latent variable models for visualisation of high dimensional data," *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 329–336, 2004.

[78] N. D. Lawrence, *Gaussian Process Models for Visualisation of High Dimensional Data*, PhD thesis, University of Cambridge, 2000.

[79] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An Introduction to Variational Methods for Graphical Models," *Machine Learning*, vol. 37, pp. 183–233, 1999.

[80] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[81] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[82] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive Auto-Encoders: Explicit Invariance During Feature Extraction," in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pp. 833–840, 2011.

[83] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

[84] Lee, J. M. (2018). *Introduction to Riemannian Manifolds* (2nd ed.). Springer.

[85] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning Internal Representations by Error Propagation*. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pp. 318–362. MIT Press.

[86] Baldi, P., & Hornik, K. (1989). *Neural networks and principal component analysis: Learning from examples without local minima*. Neural Networks, 2(1), 53–58.